Here is a general overview of the various file transfer scenarios within the NAS environment, with pointers to related articles.

File Transfers Between NAS Hosts

For basic information about transferring files within the NAS secure enclave, see the following articles:

- Local File Transfer Commands - cp, cxfscp, mcp, shiftc, rsync
- Remote File Transfer Commands - scp, shiftc
- Pleiades Front-End Usage Guidelines - file transfer between the compute systems or mass storage systems
- Streamlining File Transfers from Pleiades Compute Nodes to Lou - within a PBS job
- Checking File Integrity - to ensure the integrity of the data before and after the transfer

File Transfers Between a NAS Host and Your Local Host

Transferring files between a NAS host (such as Pleiades, Endeavour, or Lou) and a remote host, such as your local desktop, is more complex. There are multiple factors that you should be aware of:

Which Commands to Use

Remote file transfer commands such as scp and sup/shiftc are supported on most NAS systems. Depending on the way the transfers are performed, you may need one or both of the client and server software for SCP and/or the SUP/Shift client installed on your local host.

Transfer Rates

File transfer rates with the scp command, especially using SCP from versions of OpenSSH older than 4.7, can be as slow as 2 megabytes per second (MB/sec). For transferring large files over a long distance, consider doing the following:

- Upgrade to the the latest version of OpenSSH.
- Enable compression by adding -C to the scp command line if the data will compress well.
- Use sup/shiftc.

Security Issues

- With SCP, your authentication information (such as password or passcode) and data are encrypted.
- With shiftc without the --secure option, only the authentication information is encrypted, while data is not.
- You can use the SFEs or Secure Unattended Proxy (SUP). SUP is the easiest way to transfer files from and/or to your site if your local system is configured to allow the transfer.

Inbound File Transfers

When a file transfer command is initiated on a remote host, such as your local desktop, the transfer must go through either the SFEs or SUP.

Using the Secure Front Ends

Going through the SFEs requires authentication via RSA SecurID at the time of the transfer. You will be prompted for your RSA SecurID passcode when you issue the file transfer commands (such as scp).

You can use one of the following approaches:

1. One step via SSH Passthrough: Initiate scp from your local system to Pleiades, Endeavour, or Lou if SSH Passthrough has been set up.
2. One step via SSH ProxyCommand: Initiate scp from your local system through an SFE using the ssh -oProxyCommand option.

To learn more, see Inbound File Transfers through SFEs Examples.

Using the Secure Unattended Proxy

Going through the SUP does *not* require RSA SecurID token authentication at the time of transfer. Instead, special "SUP keys" using RSA SecurID authentication must be obtained ahead of time. The SUP keys are good for one week and are used automatically to authenticate your file transfers using scp or shiftc issued on a command line or in a job script.

TIP: We highly recommended that you learn about and use the Shift tool, which can be used together with the SUP to provide automated, reliable, and fast file transfers.
WARNING: Although users have accounts on the SUP servers, no login session is allowed.

File transfers going through SUP offers multiple benefits over going through the SFEs:

- SUP allows transfers to be unattended; that is, you do not have to type in your password, passphrase, or passcode when the file transfer command is issued. So, file transfers can be done within a script that can be scheduled to run ahead of time. File transfers through the SFEs cannot be done in a script.
- File transfers through SUP are done in one step, and setting up SSH passthrough is not needed since the SFEs are not involved.

See Using the Secure Unattended Proxy (SUP) and Shift Transfer Tool Overview for more information.

NAS Username and Your Local Username

If your NAS username and local username are different, you may have to add the appropriate username in the scp or sup/shiftc command line.

- If you issue the command on your local host, then *username* is your NAS username.
- If you issue the command on a NAS host, then *username* is your local username.

In the examples shown in Outbound File Transfer Examples and Inbound File Transfers through SFEs Examples, you will find the correct syntax for adding the appropriate username in the file transfer commands.

For inbound file transfers, if you have correctly included your NAS username in the ~/.ssh/config file of your local system, you do not have to include your NAS username in the scp or sup/shiftc command. A template for ~/.ssh/config is available for download.

Improving File Transfer Performance

Some file transfer commands provide options that can be used to improve your transfer rates. For example, enabling compression during file transfers may help in some cases; with the shiftc command, you can use multiple streams instead of a single stream for better performance. Read Tips for File Transfers and Increasing File Transfer Rates for more information.

File transfer performance is also dependent on some system-wide settings. If necessary, ask your local system administrator to look into issues discussed in the following articles:

- TCP Performance Tuning for WAN Transfers
- Optional Advanced Tuning for Linux
- Pittsburgh Supercomputing Center's Enabling High Performance Data Transfers - a properly tuned TCP/IP stack

The NAS Networks team can help you analyze the performance your file transfers. Contact us at support@nas.nasa.gov.

It is a good practice to confirm whether files are copied correctly to the Lou mass storage system after you transfer them.

TIP: The easiest way to verify the integrity of your file contents is to use the NAS-developed Shift tool to transfer the files. By default, Shift automatically performs a checksum operation on the data at both the source and the destination, as part of the transfer. If corruption is detected, partial file transfers and checksum operations will be performed until the problem is fixed.

If you transfer files using a command other than Shift, such as scp, the simplest and most lightweight method to verify transferred files is to compare the size (disk usage) and number of files of the original with that of the copy. For example, if you use scp to transfer *dir1* from pfe21 to lfe5:

```
pfe21% du -sk dir1
353760  dir1
pfe21% find dir1 | wc -l
51

pfe21% scp -rp dir1 lfe5:

lfe5% du -sk --apparent-size dir1
353684  dir1
lfe5% find dir1 | wc -l
51
```

In the example, the directory sizes nearly match (353760; 353684), and the number of files matches exactly (51).

Note: In most cases the sizes will not match exactly. Using the--apparent-size option of the du command is necessary on the Lou systems because the data may reside on either tape or disk.

Using Shift for Local and Remote Transfers (Recommended)

Shift Transfer Tool Overview

The NAS-developed Shift tool can copy files locally on NAS enclave hosts, transfer files between hosts inside the NAS enclave, and transfer files between the NAS enclave and remote hosts. You can also use Shift to check the status of transfers at any time, receive email notification of completion, errors, and warnings, and restart interrupted transfers or transfers with errors.

All functionality is accessed through the Shift client, which is invoked via the `shiftc` command. The syntax for `shiftc` is similar to the syntax for the cp and scp commands.

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Advanced Features

Shift includes the following advanced features:

- Automatic parallelization of transfers
- Local and remote tar creation and extraction
- Synchronization based on modification times and checksums (similar to `rsync`)
- Automatic file integrity verification and correction
- Ability to restart transfers
- Automatic retrieval of files from tape storage (DMF-managed Lou filesystems)
- Ability to check status of current transfers

How to Use Shift

See the following articles for detailed information about how to use Shift:

- Using Shift for Local Transfers and Tar Operations
- Using Shift for Transfers and Tar Operations Between Two NAS Hosts
- Using Shift for Remote Transfers and Tar Operations
- Checking Shift Transfer Status and Restarting Transfers
- Shift Command Options

Additional Resources

You can also see presentation slides for three HECC training webinars that demonstrate how to use the tool:

- Simplifying and Optimizing Your Data Transfers (PDF)
- Advanced Features of the Shift Automated File Transfer Tool (PDF)
- Simple Automated File Transfers Using SUP and Shift (PDF)

Recordings of each presentation are also available in the Past Webinars Archive.

Note: Some hostnames and options may have changed since the webinars were presented.

Shift can provide status on transfers currently in progress, and can report results of transfers that were performed within the past week. You can restart and complete any interrupted transfers or transfers that completed with errors.

Note: Transfers that have been inactive for more than a week cannot be restarted.

- Check transfer status from a NAS high-end computing host (for example, pfe*21*):

  pfe*21*% shiftc --status

- Check transfer status from remote host (for example, your local system):

  your_local_system% sup shiftc --status

- Restart a transfer that has errors or was interrupted:

  pfe*21*% shiftc --restart --id=*transfer_id*

Checking Transfer Status

Show the status of all transfers:

```
pfe21% shiftc --status
id | state | dirs | files |     file size |  date |    run |    rate
   |       | sums | attrs |      sum size |  time |   left |
---+-------+------+-------+---------------+-------+--------+---------
 1 | done  |  0/0 |   1/1 |     92KB/92KB | 10/03 |     2s |   46KB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:06 |        |
 2 | done  |  0/0 |   1/1 |     92KB/92KB | 10/03 |     8s | 11.5KB/s
   |       |  0/0 |   1/1 |     0.0B/0.0B | 17:06 |        |
 3 | done  |  1/1 |   2/2 |     99KB/99KB | 10/03 |     1s |   99KB/s
   |       |  4/4 |   0/0 |   198KB/198KB | 17:07 |        |
 4 | error |  1/1 |   1/2 |     92KB/99KB | 10/03 |     3s | 30.7KB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:08 |        |
 5 | done  |  1/1 | 64/64 | 65.5GB/65.5GB | 10/03 |    29s | 2.26GB/s
   |       |  0/0 |   0/0 |     0.0B/0.0B | 17:09 |        |
```

Showing Detailed Status

The examples in this section show how to display details about specific transfers that are listed in the output in the previous example.

TIP: Showing the detailed status of a single transfer might result in extremely large output, as a single transfer may contain hundreds, thousands, or even millions of transferred files—and multiple lines may be output for each file. Therefore, if you want to see the detailed status of a very large transfer, redirect the output to a file. For example:

```
shiftc --status --id=5 > status.id5
```

You can also search for files matching a given pattern, and show only their status. For example:

```
shiftc --status --id=1 --search=file2
```

Show the detailed status of all operations in transfer #2, from your local system:

```
your_local_system% sup shiftc --status --id=2
state | op    | target              | size |  date |    run |   rate
      | tool  | info                |      |  time |   left |
------+-------+---------------------+------+-------+--------+-------
done  | cp    | lfe2:/u/user1/file1 | 92KB | 10/03 |     5s | 18KB/s
      | rsync | -                   |      | 17:06 |        |
done  | chattr | lfe2:/u/user1/file1 |    - | 10/03 |     1s |      -
      | sftp  | -                   |      | 17:06 |        |
```

Show the detailed status of all operations in transfer #3 that involve a filename containing *file2*:

```
pfe21% shiftc --status --id=3 --search=file2
state | op    | target              | size |  date |    run |  rate
      | tool  | info                |      |  time |   left |
------+-------+---------------------+------+-------+--------+------
done  | cp    | /username/dir2/file2 |  7KB | 10/03 |     1s | 7KB/s
      | mcp   | -                   |      | 17:07 |        |
done  | cksum | /username/dir2/file2 |  7KB | 10/03 |     1s | 7KB/s
      | msum  | -                   |      | 17:07 |        |
```

Show the detailed status of all operations in transfer #4 that have an error state:

```
pfe21% shiftc --status --id=4 --state=error
state | op    | target                    | size |  date |    run | rate
      | tool  | info                      |      |  time |   left |
------+-------+---------------------------+------+-------+--------+-----
error | cp    | /username/dir2/file2      |  7KB |     - |      - |    -
      | rsync | rsync: send_files         |      |       |        |
      |       | failed to open            |      |       |        |
      |       | "/username/dir2/file2":   |      |       |        |
      |       | Permission denied         |      |       |        |
```

Showing Transfer History

Show the history of all transfers:

```
pfe21% shiftc --history
id | origin              | command
---+---------------------+-------------------------------------------------------------------------
 1 | pfe21.nas.nasa.gov | shiftc file1 /u/username/dir1
   | [/u/user1]          |
 2 | pfe21.nas.nasa.gov | shiftc -p file1 lfe2:
   | [/u/user1]          |
 3 | your_local_system   | sup shiftc -r --no-verify /username/dir1 lfe2:/username/dir2
   | [/Users/user1]      |
 4 | your_local_system   | sup shiftc -r --secure lfe2:/username/dir2 .
   | [/username]         |
 5 | pfe21.nas.nasa.gov | shiftc -r --hosts=2 /nobackup/user1/bigdir1 /nobackup/user1/bigdir2
   | [/u/user1]          |
```

Show the history of all transfers that involve a host or a command containing lfe2 from your local system:

```
your_local_system% sup shiftc --history --search=lfe2
id | origin              | command
---+---------------------+-------------------------------------
 2 | pfe21.nas.nasa.gov | shiftc -p file1 lfe2:
   | [/u/user1]          |
 4 | your_local_system   | shiftc -r --secure lfe2:/username/dir2 .
   | [/username]         |
```

Shift Command Options

Options for the shiftc command are described in this section. For more information, including syntax and examples, see **man shiftc** on any Pleiades or Lou front-end system (PFE or LFE).

Initialization Options

Transfers are initialized using syntax identical to cp and scp commands for local and remote transfers, respectively.

--clients=NUM
: Parallelize the transfer by using additional clients on each host. If the number given is 1, no additional clients will be used. A number greater than 1 will fork additional processes on each host to more fully utilize system resources and improve transfer performance.

--create-tar
: Create a tar file of all sources at the destination, which must be a non-existing filename. This option implies --recursive and --no-offline. By default, multiple tar files are created at 1 TB boundaries. The split size may be changed or splitting disabled using the --split-tar option. The --index-tar option may be used to produce a table of contents file for each tar file created. Note that this option cannot be used with --sync.

Create any missing parent directories. This option allows files to be transferred to a directory hierarchy that may not already exist, similar to the -d option of the install command.

-L, --dereference
: Always follow symbolic links to both files and directories. Note that this can result in file and directory duplication at the destination as all symbolic links will become real files and directories.

-d, --directory
: Create any missing parent directories. This option allows files to be transferred to a directory hierarchy that may not already exist, similar to the -d option of the install command.

--exclude=REGEX
: Do not transfer source files matching the given regular expression. Note that regular expressions must be given in Perl syntax (for details, see perlre(1) on The Perl Foundation website) and should be quoted on the command line when including characters normally expanded by the shell (such as "*"). Shell wildcard behavior can be approximated by using ".*" in place of "*".

--extract-tar
: Extract all source tar files to the destination, which must be an existing directory or non-existing directory name. This option implies --no-offline. Note that only tar archives in the POSIX ustar format are supported, but GNU extensions for large UIDs, GIDs, file sizes, and filenames are handled appropriately. Also note that this option cannot be used with --sync.

-f, --force
: Overwrite existing read-only files at the destination by temporarily adding owner write permission. File permissions will be restored later in the transfer. Note, however, that if the transfer does not complete successfully, files may be left with the wrong permissions. Also note that files marked as immutable using chattr +i cannot be overwritten even when this option is in effect.

--host-file=FILE
: Parallelize the transfer by using additional clients on the hosts specified in the given file (one hostname per line). This option implies a value for the --hosts option that is equal to the number of hosts in the file plus any additional hosts from the --host-list option. Fewer hosts may be used by explicitly specifying a value for the --hosts option. Note that the actual number of client hosts used will depend on the number of hosts that have equivalent access to the source and/or destination filesystems. Within PBS job scripts, this option can be set to the $PBS_NODEFILE variable to use all nodes of the job.

--host-list=LIST
: Parallelize the transfer by using additional clients on the hosts specified in the given comma-separated list. This option implies a value for the --hosts option that is equal to the number of hosts on the list plus any additional hosts from the --host-file option. Fewer hosts may be used by explicitly specifying a value for the --hosts option. Note that the actual number of client hosts used will depend on the number of hosts that have equivalent access to the source and/or destination filesystems.

--hosts=NUM
: Parallelize the transfer by using additional clients on (at most) the given number of hosts. If the number given is 1, no additional clients will be used. A number greater than 1 enables automatic transfer parallelization where additional clients may be invoked on additional hosts to increase transfer performance. Note that the actual number of clients used will depend on the number of hosts for which Shift has filesystem information and the number of hosts that have equivalent access to the source and/or destination filesystems. Client hosts will be accessed as the current user with host-based authentication or an existing ssh agent that contains an ssh from a file matching ~/.ssh/id*.

--identity=FILE
: Authenticate to remote systems using the given ssh identity file. The corresponding public key must reside in the appropriate user's ~/.ssh/authorized_keys file on the remote host. Note that only identity files without passphrases are supported. If a passphrase is required, an ssh agent may be used instead, but with a loss of reliability. This option is not needed if the remote host accepts host-based authentication from client hosts.

-I, --ignore-times
: By default, the --sync option skips the processing of files that have the same size and modification time at the source and destination. This option specifies that files should always be processed by checksum regardless of size and modification time.

--include=REGEX
: Only transfer source files matching the given regular expression. Note that regular expressions must be given in Perl syntax (see perlre(1) on the Perl Foundation website for details) and should be quoted on the command line when including characters normally expanded by the shell (such as "*"). Shell wildcard behavior can be approximated by using ".*" in place of "*".

--index-tar
: Create a table-of-contents file for each tar file created with the --create-tar option. The table of contents will show each file in the tar file along with permissions, user/group ownership, and size. For a tar file "file.tar", the table of contents will be named "file.tar.toc". Unless the --no-verify option is used, a checksum file named "file.tar.sum" will also be created, which is suitable as input for msum --check-tree -c. Note that when the --split-tar option is used, multiple table-of-contents files may be created. For each split tar file "file.tar-i.tar", the table of contents will be named "file.tar--.tar.toc" and the checksum file will be named "file.tar-i.tar.sum".

--newer=[TYPE:]DATE
: Only transfer source files whose modification time (or combination of modification, access, and/or creation times) is newer (inclusive) than the given date. Any date string supported by the Perl Date::Parse module (see Date::Parse(3) for details) can be specified. An optional type expression of the form "[acmACM]+(|[acmACM]+)*)" can be given to specify conditions in which one or more conditions are or are not newer than the date, where: "a" is access time; "c" is creation time; "m" is modification time; and "A", "C", and "M" are their inverses, respectively. For example, "aM|cm" would transfer source files whose access time was newer than the date but whose modification time was not newer, or files whose creation time and modification time were newer. Note that this option can be combined with --older to specify exact date ranges.

-P, --no-dereference
: Never follow symbolic links to files or directories. Note that this can result in broken links at the destination, as files and directories referenced by symbolic links that were not explicitly transferred or implicitly transferred using --recursive might not exist on the target.

-T, --no-target-directory
: Do not treat the destination specially when it is a directory or a symbolic link to a directory. This option can be used with recursive transfers to copy a directory's contents into an existing directory instead of into a new subdirectory beneath it as is done by default.

--older=[TYPE:]DATE
: Only transfer source files whose modification time (or combination of modification, access, and/or creation times) is older than the given date. Any date string supported by the Perl Date::Parse module (see Date::Parse(3) for details) can be specified. An optional type expression of the form "[acmACM]+(|[acmACM]+)*)" can be given to specify conditions in which one or more conditions are or are not older than the date, where: "a" is access time; "c" is creation time; "m" is modification time; and "A", "C", and "M" are their inverses, respectively. For example, "aM|cm" would transfer source files whose access time was older than the date but whose modification time was not older, or files whose creation time and modification time were both older. Note that this option can be combined with --newer to specify exact date ranges.

--pipeline
: Produce verified files earlier in the transfer by preferring to process the normal sequence of operations (find, copy, checksum, verify checksum, change attributes) in reverse order. In default (non-pipeline) operation, these stages are performed in order where all files are found before any are copied; before any are checksummed, etc. When this option is enabled, files that have reached the change attribute stage will be processed before files that have reached the verify checksum stage, which will be processed before files that have reached the checksum stage, etc. This allows you to perform parallel processing on verified files while the transfer is still ongoing. To determine the list of files that have been successfully verified in a transfer with id "N", use --status --id=N --state=done --search=chattr. When multiple clients are participating in the transfer (i.e., --clients or --hosts options are specified with a value greater than 1), different clients will prefer different stages for more overlap of reads and writes between the source and destination filesystems.
  Note that while several strategies are employed to ensure that checksums are computed from disk and not from cache, it is safest to use this option only when there is actually a need to process destination files during the transfer.

--ports=NUM1:NUM2
: Use ports from the range NUM1-NUM2 for the data streams of TCP-based transports (currently bbftp and fish-tcp). All connections originate from the client host so the given port range must be allowed on the network path to the remote host and by the remote host itself.

-R, -r, --recursive
: Transfer directories recursively. This option implies --no-dereference. Note that any symbolic links pointing to directories that are given on the command line will be followed during recursive transfers (identical to the default behavior of the cp command).

--secure
: Encrypt data during remote transfers and use secure ciphers and MACs with SSH-based transports. Note that this option will, in most cases, decrease performance as it eliminates some higher performance transports and increases CPU utilization during SSH connections.

--sync
: Synchronize files between the source and destination, similar to the rsync command. By default, files that have the same size and modification time at the source and destination will not be transferred. If the size or modification time of a file differs between the two, the contents of the file will be compared via checksum and any portions that differ will be transferred to the destination. To skip the size and modification time checks and always begin with the checksum stage, use -I or --ignore-times. If the --no-verify option is specified, integrity verification is not performed; this will increase performance when there are many files at the source that are not at the destination, but will decrease performance when there are large files that have only small changes between the source and destination. Setting the --retry option to zero with this option can be used to show which files differ without making any changes. Note that when syncing directories, the destination should be specified as the parent of the location where the source directory should be transferred to. Also note that this option cannot be used with the --create-tar or --extract-tar options.

--user=USER
: Set the user that will be used to access remote systems.

--wait
: Block until the transfer completes and print a summary of the transfer. This option implies --no-mail. An exit value of 0 indicates that the transfer has successfully completed while an exit value of 1 indicates that the transfer has failed or that the waiting process was terminated prematurely. This option may be used together with --monitor to show the real-time status of the transfer while waiting.

Feature Disablement Options

The following options disable certain default features.

--no-cron
: Do not attempt to recover from host/process failures via cron. Note that when such a failure occurs, the transfer will become stuck in the "run" state until stopped.

--no-mail[=LIST]
: By default, emails are sent when a transfer completes successfully, aborts with errors, or is stopped; and for the first instances of alerts, errors, throttling, and/or warnings while running. This option prevents emails from being sent altogether or, optionally, for a specific subset of states. The given list may be a comma-separated subset of {alert, done, error, run, stop, throttle, warn}. This option may be desirable when performing a large number of scripted transfers. Note that equivalent transfer status and history information can always be manually retrieved using the --status and --history options, respectively.

--no-offline
: By default, files transferred to and from DMF-managed filesystems will be migrated to offline media as soon as the transfer is completed. This option specifies that files should instead be kept online (not migrated). Note that DMF may still choose to migrate a file even when this option is enabled.

--no-preserve[=LIST]
: By default, times, permissions, ownership, striping, ACLs, and extended attributes of transferred files and directories are preserved when possible. This option specifies that these items (or an optional specified subset) should not be preserved. The given list may be a comma-separated subset of {acl, mode, owner, stripe, time, xattr}. Note that permissions may be left in various states depending on the invoking user's umask and the transport utilized. In particular, read access at the destination may be more permissive than read access at the source.

--no-recall
: By default, files transferred from DMF-managed filesystems will be recalled from offline media as soon as the transfer begins and again before each batch of files is processed. This option specifies that files should not be recalled. Note that DMF will still recall files as needed even when this option is enabled.

--no-sanity
: Disable file existence and size checks at the end of the transfer. This option was included for benchmarking and completeness purposes and is not recommended for general use.

--no-silent
: By default, the checksums of all files transferred with Shift are stored in a per-user database. When a file with a known checksum is transferred and has not been modified since the checksum was stored, the transfer will be put into the "alert" state if the current checksum does not match the stored checksum. This option disables the storage of checksums and comparison against existing checksums. While silent corruption detection adds minimal overhead during normal operation, it can increase the probability of lock contention when there are large numbers of clients.

--no-verify
: By default, files are checksummed at the source and destination to verify that they have not been corrupted and if corruption is detected, the corrupted portion of the destination file is automatically corrected using a partial transfer from the original source. This functionality decreases the performance of transfers in proportion to the file size. If assurance of integrity is not required, the --no-verify option may be used to disable verification.

History, Management, and Status Options

Once one or more transfers have been initialized, you may view transfer history, stop/restart transfers, and/or check transfer status with the following options.

--history[=csv]
: Show a brief history of all transfers including the transfer identifier, the origin host/directory, and the original command. When --history=csv is specified, history is shown in CSV format.

--id=NUM
: Specify the transfer identifier to be used with management and status commands.

--last-sum
: Query the silent corruption database for all files given on the command line and print (one file per line) the last known checksum, the file modification time associated with this checksum, and the filename. When --index-tar is given, the first file argument is assumed to be a tar file and the remaining arguments names of files within the tar for which checksum information will be printed. A checksum of "-" means that no information is stored for the file.

--mgr=HOST
: Set the host that will be used to manage transfers. By default, this host will be accessed as the current user with host-based authentication or an existing ssh agent. The user and/or identity used to access the manager host may be changed with the --mgr-user and --mgr-identity options, respectively.

--mgr-identity=FILE
: Authenticate to the manager host using the given ssh identity file. The corresponding public key must reside in the appropriate user's ~/.ssh/authorized_keys file on the manager host. Note that only identity files without passphrases are supported. If a passphrase is required, an ssh agent may be used instead, but with a loss of reliability. This option is not needed if the manager host accepts host-based authentication from client hosts.

--mgr-user=USER
: Set the user that will be used to access the manager host. Note that if the transfer is initiated by root and the --mgr-identity option is not specified, manager communication will be performed as the given user, so that user must be authorized to run processes locally. In particular, care should be taken on PBS-controlled nodes, where the given user should either own the node or be on the user exception list.

--monitor[=FORMAT]
: Show the realtime status of all running transfers including the transfer identifier, the current state, the number of directories completed, the number of files transferred, the number of files checksummed, the number of attributes preserved, the amount of data transferred, the amount of data checksummed, the time the transfer started, the duration of the transfer, the

estimated time remaining in the transfer, and the rate of the transfer. Note that updates are realtime with respect to the information available to the manager and not with respect to the transports that may be carrying out the transfer. Status will be returned in CSV format when the --monitor=csv is specified. Duration and estimated time will be zero-padded when--monitor=pad is specified. When --monitor=color is specified, transfers in the {error, run, throttle, warn} states will be shown with {red, green, magenta, yellow} coloring, respectively. When --id is specified, only the given transfer will be shown. When all transfers (or the one specified) have completed, the command will exit. This option may be used with the --wait option to monitor progress while waiting.

--plot=[=[BY[:/]]LIST]

Produce output suitable for piping into gnuplot (version 5 or above) that shows detailed performance over time across all transfers. The --id and --state options may be used to plot only a single transfer or transfers in a particular state, respectively. The default plot will show the aggregate performance of each I/O operation (i.e. cp, sum, and cksum) and the aggregate performance of each metadata operation (i.e. find, mkdir, ln, and chattr) across all of the user's transfers. Operations and/or additional groupings are shown on the left y-axis axis across time on the x-axis with heat-based coloring indicating MB/s for I/O operations or operations per second for metadata operations. In addition, aggregate I/O and metadata performance will be shown as an overlaid point plot with green and blue points, respectively.

The list of plotted items may be changed by giving a comma-separated list consisting of one or more of the stages {chattr, cksum, cp, find, io, ln, meta, mkdir, sum} and/or one or more of the tools {bbftp, fish, fish-tcp, mcp, msum, rsync, shift-cp, shift-sum}. Note that "io" is a shorthand for "cp,sum,cksum", "meta" is a shorthand for "find,mkdir,ln,chattr", and "tool" is a shorthand for "bbftp,fish,fish-tcp,mcp,msum,rsync,shift-cp,shift-sum".

The list of items may be grouped by any of {client, fs, host, id, net, user} by prefixing one of these terms to the list. For example, --plot=id:cp would show a plot of the copy performance achieved by each transfer id. When a grouping is given without a specific list of metrics (e.g., --plot=id), "io" is assumed. When a slash "/" is used instead of colon ":", a heatmap-based bubble plot will be created with the size of each circle indicating the relative size of the batch of operations. For example, --plot=fs/tool would show a plot of the performance that each tool achieved on each file system with relative batch size.

--restart[=ignore]

Restart the transfer associated with the given --id that was stopped due to unrecoverable errors or stopped explicitly via the --stop option. If --restart=ignore is specified, all existing errors will be ignored and the transfer will progress as if the associated files and directories were no longer part of the transfer. Note that transfers must be restarted on the original client host or one that has equivalent filesystem access. A subset of the available command-line options may be re-specified during a restart, including --bandwidth, --buffer, --clients, --cpu, --disk, --files, --force, --host-file, --host-list, --hosts, --interval, --io, --ior, --iow, --local, --net, --netr, --netw, --no-cron, --no-mail, --no-offline, --no-recall, --no-silent, --pipeline, --ports, --preallocate, --remote, --retry, --secure, --size, --streams, --stripe, --threads, and --window.

--search=REGEX

When the --status and --id options are specified, this option will show the full status of file operations in the associated transfer whose source or destination filename match the given regular expression. When the --history option is specified, this option will show a brief history of the transfers in which the origin host or original command matches the given regular expression. Note that regular expressions must be given in Perl syntax (see perlre(1) for details).

--state=STATE

When the ??status and ??id options are specified, --state=STATE will show the full status of file operations in the associated transfer that have the given state. When --id is not specified, this option will show the brief status of transfers in the given state. Valid states are done, error, none, queue, run, and warn. A state of "none" will show a summary of the given transfer.

--stats[=csv]

Show stats across all transfers including transfer counts, rates, tool usage, initialization options, error counts, and error messages. When --stats=csv is specified, stats are shown in CSV format without error messages.

--status[=FORMAT]

Show a brief status of all transfers including the transfer identifier, the current state, the number of directories completed, the number of files transferred, the number of files checksummed, the number of attributes preserved, the amount of data transferred, the amount of data checksummed, the time the transfer started, the duration of the transfer, the estimated time remaining in the transfer, and the rate of the transfer. When the number of transfers exceeds a set threshold (the default is 20), older successfully completed transfers beyond that limit will be omitted for readability. These omitted transfers can be shown using --status with --state=done. Status will be returned in CSV format when --status=csv is specified. Duration and estimated time will be zero-padded when --status=pad is specified. When --status=color is specified, transfers in the {done, error, run, stop, throttle, warn} states will be shown with {default, red, green, cyan, magenta, yellow} coloring, respectively.

When ??id is specified, --status[=FORMAT] will show the full status of every file operation in the associated transfer. For each operation, this includes the state, the type, the tool used for processing, the target path, associated information (error messages, checksums, byte ranges, and/or running host) when applicable, the size of the file, the time processing started, and the rate of the operation. Note that not all of these items will be applicable at all times (for example, "rate" will be empty if the state is "error"). Also note that operations are processed in batches so the rate shown for a single operation will depend on the other operations processed in the same batch. When --status=color is specified, operations in the {done, error, queue, run, warn} states will be shown with {default, red, cyan, green, yellow} coloring, respectively.

--stop

Stop the transfer associated with the given --id. Note that transfer operations currently in progress will run to completion but new operations will not be processed. Stopped transfers may be restarted with the --restart option.

Transfer Tuning Options

Some advanced options are available to tune various aspects of the shiftc command's behavior. These options are not needed by most users.

--bandwidth=BITS

Choose the TCP window size and number of TCP streams of TCP-based transports (currently, bbftp and fish-tcp) based on the given bits per second. The suffixes k, m, g, and t may be used for Kb, Mb, Gb, and Tb, respectively. The default bandwidth is estimated to be 10 Gb/s if a 10 GE adapter is found on the client host, 1 Gb/s if the client host can be resolved to an

organization domain (by default, one of the six original generic top-level domains), and 100 Mb/s otherwise.

--buffer=SIZE

Use memory buffer(s) of the given size when configurable in the underlying transport being utilized (currently, all butrsync). The suffixes k, m, g, and t may be used for KiB, MiB, GiB, and TiB, respectively. The default buffer size is 4 MiB. Increasing the buffer size trades higher memory utilization for more efficient I/O.

--files=COUNT

Process transfers in batches of at least the given number of files. The suffixes k, m, b or g, and t may be used for 1E3, 1E6, 1E9, and 1E12, respectively. The default batch count is 1000 files. This option works in concert with --size and --interval to manage the number of checkpoints and the overhead of transfer management. A batch will initially consist of at least --files files or --size bytes, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. To make batch selection completely dynamic, use --files=1 and --size=1.

--interval=SECS

Process transfers in batches that take about the given number of seconds. The default interval is 30 seconds. This option works in concert with --files and --size to manage the number of checkpoints, as well as the overhead of transfer management. A batch will initially consist of at least --files files or --size bytes, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. Note that the actual time a batch takes will depend on its contents, and that the interval will be increased as the number of clients participating in a transfer increases to minimize contention for manager locks. To make batch selection completely static, use --interval=0.

--local=LIST

Specify one or more local transports to be used for the transfer in order of preference, separated by commas. Valid transports for this option currently include bbftp, cp, fish, fish-tcp, mcp, and rsync. Note that the given transport(s) will be given priority, but may not be used in some cases (for example, rsync is not capable of transferring a specific portion of a file as needed by verification mode). In such cases, the default transport based on File::Copy will be used. The tool actually used for each file operation can be shown using the --status option with the --id option set to the given transfer identifier.

--preallocate=NUM

Preallocate files when their sparsity is under the given percent, where sparsity is defined as the number of bytes a file takes up on disk divided by its size. Note that this option will only have an effect when the fallocate command is available, the destination file does not already exist, and the target filesystem properly supports fallocate's -n option. Also note that this option will not function properly when either bbftp or rsync (to a DMF filesystem) is utilized as the transport due to their use of temporary files.

--remote=LIST

Specify one or more remote transports to be used for the transfer in order of preference, separated by commas. Valid transports for this option currently include bbftp, fish, fish-tcp, rsync, and sftp. Note that the given transport(s) will be given priority, but may not be used in some cases (for example, bbftp is not capable of transferring files with spaces in their names and is also incompatible with the --secure option). In such cases, the default transport based on sftp will be used. The tool actually used for each file operation can be shown using --status with the --id option set to the given transfer identifier.

--retry=NUM

Retry operations deemed recoverable up to the given number of attempts per file. The default number of retries is 2. A value of zero disables retries. Note that disabling retries also disables the ability of --sync to change file contents. Also note that the given value is cumulative across all stages of a file's processing so different stages may not be retried the same number of times.

--size=SIZE

Process transfers in batches of at least the given total file size. The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default batch size is 4 GB. This option works in concert with --files and --interval to manage the number of checkpoints and the overhead of transfer management. A batch will initially consist of at least --size bytes or --files files, whichever is reached first. The batch may then be dynamically increased in size until there is enough work to span --interval seconds. To make batch selection completely dynamic, use --files=1 and --size=1.

--split=SIZE

Parallelize the processing of single files using chunks of the given size. The suffixes k, m, g, and t may be used for KiB, MiB, GiB, and TiB, respectively. The default split size is zero, which disables single file parallelization. A split size of less than 1 GiB is not recommended. Lowering the split size will increase parallelism but decrease the performance of each file chunk and increase the overhead of transfer management. Raising the split size will have the opposite effect. The ideal split size for a given file is the size of the file divided by the number of concurrent clients available. Note that --split=SIZE does not have an effect unless the value of the --hosts option is greater than 1. Also note that --split=SIZE can, in some cases, decrease remote transfer performance as it eliminates some higher performance transports.

--split-tar=SIZE

Create tar files of around the given size when used with the --create-tar option. When multiple tar files are created for a destination tar file "file.tar", the resulting split tar files will be named "file.tar-i.tar" starting from "file.tar-1.tar". The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default split tar size is 1 TB. A value of zero disables splitting. A split tar size of greater than 2 TB is not recommended. Note that resulting tar files may still be larger than specified when source files exist that are larger than the given size.

--streams=NUM

Use the given number of TCP streams in TCP-based transports (currently bbftp and fish-tcp). The default is the number of streams necessary to fully utilize the specified/estimated bandwidth using the maximum TCP window size. Note that it is usually preferable to specify the --bandwidth option, which allows an appropriate number of streams to be set automatically. Increasing the number of streams can increase performance when the maximum window size is set too low or there is cross-traffic on the network, but too high a value can decrease performance due to increased congestion and packet loss.

--stripe=[CEXP][::[SEXP][::PEXP]]

By default, a file transferred to a Lustre filesystem will be striped according to an administrator-defined policy (one stripe per GiB when not configured). It is recommended, although not required, that this policy preserve existing striping when the source resides on Lustre and has non-default striping. To disregard existing striping, "stripe" may be used with the --no-preserve=stripe option. To disable automatic striping completely and use the default Lustre behavior for all files and directories, use --stripe=0.

The user may override the default policy by specifying expressions for one or more of the stripe count (CEXP), stripe size

(SEXP), and stripe pool (PEXP). For the stripe count, a positive number less than 65,536 indicates a fixed number of stripes to use for all destination files and directories. A greater number or size defined with the suffixes k, m, g, and t for KiB, MiB, GiB, and TiB, respectively, specifies that files will be allocated one stripe per given size while directories will be striped according to the default policy. Finally, an arbitrary Perl expression (see perlsyn(1) for details) involving the constants NM, SZ, SC, and SS for source name, size, stripe count, and stripe size, respectively, may be specified to dynamically define the stripe count differently for every file and directory in the transfer. For example, the expression "NM =~ /foo/ ? 4 : (SZ < 10g ? 2g : 10g)" would set the stripe count of files whose name contains "foo" to 4, and the stripe count of files whose name does not contain "foo" to either one stripe per 2 GiB when the file size is less than 10 GiB or one stripe per 10 GiB otherwise.
Striping behavior may be further refined by specifying a stripe size expression and/or Lustre pool name expression with similar conventions. The stripe count and/or stripe size can be left empty before the colons when specifying the stripe size or pool, respectively. For example, --stripe=::4m would specify the stripe size to be 4 MiB while using the default stripe count policy and, similarly, --stripe=:::pool1 would use the pool "pool1" while using the default stripe count and stripe size. Note that if the stripe pool is a Perl expression and not a simple alphanumeric pool name, pool names must use Perl conventions for indicating strings such as quotes and/or quote-like operators, for example: "NM =~ /foo/ ? q(poolfoo) : q(poolbar)"

--threads=NUM
: Use the given number of threads in multi-threaded transports and checksum utilities (currently, mcp and msum). The default number of threads is 4. Increasing the number of threads can increase transfer/checksum performance when a host has excess resource capacity, but can reduce performance when any associated resource has reached its maximum.

--verify-fast
: By default, files are checksummed at the source and destination to verify that they have not been corrupted with the source being read once during the copy and again during the checksum. This option specifies that the source copy buffer should be reused when possible for the source checksum calculations. This potentially increases performance up to 33%, but does not allow bits corrupted during the initial read to be detected.

--window=SIZE
: Use a TCP send/receive window of the given size in TCP-based transports (currently, bbftp and fish-tcp). The suffixes k, m, g, and t may be used for KB, MB, GB, and TB, respectively. The default is the product of the specified/estimated bandwidth and the round-trip time between source and destination. Note that it is usually preferable to specify the ??bandwidth option, which allows an appropriate window size to be set automatically. Increasing the window size allows TCP to operate more efficiently over high bandwidth and/or high latency networks, but too high a value can overrun the receiver and cause packet loss.

Transfer Throttling Options

Transfers can be throttled to prevent resource exhaustion when they reach configured thresholds for CPU, disk, I/O, and/or network utilization.

--cpu=NUM
: Throttle the transfer when the local CPU usage reaches the specified percent of the total available. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local CPU. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average CPU utilization equal to the value specified. Note that this functionality is currently supported only on Unix-like systems.

--disk=NUM1:NUM2
: Suspend/resume the transfer when the target filesystem disk usage reaches the specified percent of the total available. This option is disabled by default but may be desirable to prevent transfers from consuming too much local or remote disk space. Once the first threshold is reached, the transfer will suspend until enough disk resources have been freed on the target to bring the disk utilization under the second threshold. Note that this functionality is currently only supported on Unix-like systems.

--io=NUM
: Throttle the transfer when the local I/O usage reaches the specified rate in MB/s. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local I/O bandwidth. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average I/O rate equal to the value specified.

--ior=NUM
: Throttle the transfer when the local I/O reads reach the specified rate in MB/s. This option is similar to the --io option, but only applies to reads.

--iow=NUM
: Throttle the transfer when the local I/O writes reach the specified rate in MB/s. This option is similar to the --io option, but only applies to writes.

--net=NUM
: Throttle the transfer when the local network usage reaches the specified rate in MB/s. This option is disabled by default but may be desirable to prevent transfers from consuming too much of the local network bandwidth. Once the given threshold is reached, a sleep period will be induced between each batch of files to achieve an average network rate equal to the value specified.

--netr=NUM
: Throttle the transfer when the local network reads reach the specified rate in MB/s. This option is similar to the --net option but only applies to reads.

--netw=NUM
: Throttle the transfer when the local network writes reach the specified rate in MB/s. This option is similar to the --net option but only applies to writes.

For transfers on the same host within the NAS enclave—for example, between two directories on pfe*21*—the syntax for shiftc is similar to the cp command.

Note: If source and destination paths are not specified on the command line, any number of source/destination combinations will be read from the standard input, stdin (one combination per line).

The syntax for local Shift transfers is as follows:

shiftc [*option*]... *source dest*
shiftc [*option*]... *source... directory*
shiftc [*option*]...

For information about Shift options, including those used in the examples in this article, see Shift Command Options.

Transferring Files Locally

The examples in this section show you how use the shiftc command to transfer files on the same NAS host.

Note: The first example includes output; subsequent examples show only the command line.

- Copy local *file1* into existing directory /u/*username*:

  pfe*21*% shiftc *file1* /u/*username*
  Shift id is 1
  Detaching process (use --status option to monitor progress)

- Copy *file1* from /u/*username* to /nobackup/*username*/*dir1* on pfe*21*:

  pfe*21*% shiftc /u/*username/file1* /nobackup/*username/dir1*

- Recursively copy the directory *inputs* inside another directory, /nobackup/*username/dir2*:

  % shiftc -r *inputs* /nobackup/*username/dir2*

  Complete the same operation with data verification turned off:

  % shiftc -r --no-verify *inputs* /nobackup/*username/dir2*

  TIP: Although using the --no-verify option can improve the speed of your transfer, it is not recommended because without data verification, the integrity of your data cannot be ensured.

- Copy local *file1* in the current directory to existing local directory /u/*username/dir1*:

  pfe*21*% shiftc *file1* /u/*username*/*dir1*

- Recursively copy local directory /nobackup/*username/dir1* to local directory /nobackup/*username/dir2* using 2 client hosts to perform the transfer:

  pfe*21*% shiftc -r --hosts=2 /nobackup/*username/dir1* /nobackup/*username/dir2*

- Recursively copy local directory nobackup/*username/dir1* to local directory *dir2*, but exclude files ending in .log:

  pfe21% shiftc -r --exclude='\.log$' nobackup/*username/dir1 /dir2*

Creating and Extracting Tar Files Locally

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by either shiftc or tar. In the same step, you can also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes (recommended).

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

The examples in this section show you how to create tar files on the same NAS host.

- Create a tar file (*dir1*.tar) of the directory /nobackup/*username/dir1* and put it in the current directory, along with a corresponding table of contents (*dir1*.tar.toc):

  pfe*21*% cd /nobackup/*username*
  pfe*21*% shiftc --create-tar --index-tar *dir1 dir1*.tar

  Note:? If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

- The Pleiades nobackup filesystems are mounted on Lou. This makes it possible to create local tar files from your nobackup filesystem directly to your Lou home directory when you are logged into Lou (lfe*5-8*).

  For example, to transfer the *dir1* directory as a tar file from your nobackup filesystem to the *data_dir* directory in your Lou home directory:

  lfe*5*% cd /nobackupp8/*username/data_dir*
  lfe*5*% shiftc --create-tar --index-tar *dir1* /u/*username/data_dir/dir1*.tar

  Note: In the above example, the first step is to change into the directory /nobackupp8/*username/data_dir*, which is the parent directory of *dir1* (the directory you are transferring). This prevents the tar operation from creating extraneous prepended directories when the tar file is extracted.

Creating Incremental Tar Files

The examples in this section show you how to create incremental tar files on the same NAS host.

- Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the /nobackup/*username/dir1* directory, and put it in the current directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

  pfe21% cd /nobackup/*username*
  pfe21% shiftc --create-tar --index-tar --newer="Jan 1 2020" *dir1 dir1*-2020.tar

- Create an incremental tar file (*dir1*-update.tar) of all files modified in the *dir1* directory of your nobackup directory after the original tar (*dir1*.tar) was successfully completed, to the *data_dir* directory in your Lou home directory:

  lfe5% cd /nobackupp/*username*/*data_dir*
  lfe5% shiftc --create-tar --index-tar --newer=`stat -c %Y /u/*username/data_dir/dir1*.tar` \
  dir1 /u/*username*/*data_dir*/*dir1*-update.tar

  Notes:

  - The shiftc command line shown above is too long to be formatted as one line, so it is broken with a backslash (\).
  - The version of stat currently deployed on the systems does not allow file creation time to be retrieved, so the above example might miss files that were modified or created while the original tar was being written.

Extracting Tar Files

The examples in this section show you how to extract files from a tar file on the same host.

- Extract the tar file *dir1*.tar on Lou directly into your /nobackup directory:

  lfe*5*% cd /nobackup/*username/data_dir*
  lfe*5*% shiftc --extract-tar /u/*username/data_dir/dir1*.tar .

- If *dir1* was over 1 TB and therefore split into multiple tar files:

  pfe21% shiftc --extract-tar /u/*username*/*data_dir*/*dir1.**tar .

- Extract the files *1g.20* through *1g.29* from *dir1*.tar to the current directory:

  pfe21% shiftc --extract-tar --include='1g\.2[0-9]' *dir1*.tar .

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.

For transfers between two host systems within the NAS enclave—for example, from a PFE to an LFE—the syntax for the shiftc command is nearly the same as that of local transfers.

Note: If source and destination paths are not specified on the command line, any number of source/destination combinations will be read from stdin (one combination per line).

shiftc [OPTION]... *source dest*
shiftc [OPTION]... source... *directory*
shiftc [OPTION]...

For information about Shift options, including those used in the examples in this article, see Shift Command Options.

Transferring Files Within the Enclave

The examples in this section show you how to transfer files between two hosts within the enclave.

Note: The first example includes output; subsequent examples show only the command line.

Perform basic host-to-host transfers (for example, /u/*username*/*file1* on pfe*21* to your data/*dir2* directory on Lou):

pfe*21*% shiftc /u/*username*/*file1* lfe:data/*dir2*
Shift id is 2
Detaching process (use --status option to monitor progress)

TIP: If you have a small transfer that will complete quickly, using the --wait option will provide a helpful indication that your transfer is complete—you'll know it's done as soon as your prompt returns. Also, if you include shiftc in a script, using the --wait option will prevent the next step in the script from starting until the transfer is complete.

The next two examples show this option.

- Copy local *file1* in the current directory to your Lou home directory while waiting for completion:

  pfe*21*% shiftc --wait *file1* lfe:

- Synchronize the local directory /u/*username*/*dir1* with the Lou directory /u/*username/dir2/dir1* while waiting for completion:

  pfe*21*% shiftc -r --sync --wait /u/*username*/*dir1* lfe:/u/*username*/*dir2*

Creating and Extracting Tar Files Within the Enclave

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by either shiftc or tar. In the same step, you can also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes (recommended).

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

This example creates a tar file (*dir1*.tar) of the *dir1* directory in /nobackup/*user1* directly into your Lou home directory, along with a corresponding table of contents (*dir1*.tar.toc):

pfe*21*% cd /nobackup/*user1*
pfe*21*% shiftc --create-tar --index-tar *dir1* lfe:*dir1*.tar

Note:? If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

Creating Incremental Tar Files

The examples in this section show you how to create incremental tar files from one host to another within the enclave.

- Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the directory /nobackup/*username/dir1* directly to your Lou home directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

  pfe21% cd /nobackup/*username*
  pfe21% shiftc --create-tar --index-tar --newer="Jan 1 2020" *dir1* lfe:*dir1*-2020.tar

- Create an incremental tar file (*dir1*-update.tar) of all files modified in the *dir1* directory of your nobackup directory after the original tar (*dir1*.tar) was successfully completed, to the *data_dir* directory in your Lou home directory:

  pfe21% cd /nobackupp/*username*/*data_dir*
  pfe21% shiftc --create-tar --index-tar --newer=`ssh lfe stat -c %Y *dir1*.tar` *dir1* lfe:*dir1*-update.tar

Note: The version of stat currently deployed on the systems does not allow file creation time to be retrieved, so the above example might miss files that were modified or created while the original tar was being written.

Extracting Tar Files

These examples show you how to extract tar files from one host to another within the enclave.

- Extract the *dir1*.tar file from your Lou home directory to the current directory:

  pfe*21*% shiftc --extract-tar lfe:*dir1*.tar .

- If *dir1* was over 1 TB and therefore split into multiple tar files, extract them from your Lou home directory to the current directory as follows:

  pfe*21*% shiftc --extract-tar lfe:'*dir1*.*tar' .

  Note: As shown in the example, quotation marks are required when using wildcards (e.g., *) with sources on a remote host.

- Extract the files *1g.20* through *1g.29* from *dir1*.tar to the current directory:

  pfe21% shiftc --extract-tar --include='1g\.2[0-9]' lfe:*dir1*.tar .

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.

Transfers between a remote system and a host system within the NAS enclave—for example, between your local system and a Pleiades Front-End node (PFE)—must be carried out using the Secure Unattended Proxy (SUP).

To use the SUP for Shift transfers to NAS systems, you must first download the SUP client, authorize one or more NAS hosts for SUP operations, and then authorize one or more NAS directories for writes. A brief summary of these steps is shown below. For a full overview, see Using the Secure Unattended Proxy.

Downloading SUP to Enable Remote Transfers

Complete the following steps to set up your system for remote transfers:

1. Download and install SUP client (one time):

   your_local_system% wget -O sup https://www.nas.nasa.gov/hecc/support/kb/file/9
   your_local_system% chmod 700 sup
   your_local_system% mv sup ~/bin

2. Authorize host for SUP operations (one time per host):

   your_local_system% ssh pfe*21*
   pfe*21*% touch ~/.meshrc

3. Authorize directories for writes (one or more times per host):

   your_local_system% ssh pfe*21*
   pfe*21*% echo /tmp >>~/.meshrc
   pfe*21*% echo /nobackup/$USER >>~/.meshrc
   pfe*21*% echo /u/$USER >>~/.meshrc

Performing Remote Transfers

For remote transfers, the shiftc syntax is similar to local transfers and transfers between two hosts within the enclave, except that you must prepend sup (the SUP client) to each shiftc command. Also, remote Shift transfers must always be initiated from the system that is external to the NAS enclave, but files may be transferred in either direction.

sup shiftc [OPTION]... *source dest*
sup shiftc [OPTION]... *source... directory*
sup shiftc [OPTION]...

For example, the following enclave-to-enclave transfer copies *file1* into the directory ~/data/*run2* on lfe5:

pfe21% shiftc /home/*username/file1* lfe*5*:~/data/*run2*

If the above example were changed into a remote transfer, it would become:

your_local_system% sup shiftc /home/*username/file1* lfe*5*:~/data/*run2*

In general, shiftc [*args*] becomes sup shiftc [*args*].

Note: For information about Shift options, including those used in the examples in this article, see Shift Command Options.

File Transfer Examples

The examples in this section show you how to transfer files between your local system and a host within the enclave. The first example includes output; subsequent examples show only the command line.

- Perform a remote transfer via the Secure Unattended Proxy (for example, /*username*/*file1* on your local system to your home directory on pfe*21*):

  your_local_system% sup shiftc /*username/file1* pfe*21*:
  Shift id is 3
  Detaching process (use --status option to monitor progress)

- Recursively copy local directory */username/dir1* on your local system to directory */username/dir2* on lfe*5*:

  your_local_system% sup shiftc -r /*username/dir1* lfe*5*:/*username/dir2*

- Recursively copy remote directory */username/dir2* on lfe*5* to the current directory on your local system using an encrypted transport:

  your_local_system% sup shiftc -r --secure lfe*5*:/*username/dir2* .

Creating and Extracting Tar Files Remotely

You can use Shift to transfer a directory and write it into a tar file it in one step, resulting in a portable tar file that can be read by either shiftc or tar. In the same step, you can also create a table-of-contents (.toc) file that lists the files contained in the archive along with their sizes and attributes (recommended).

Because of their sequential nature, tar files cannot be efficiently updated in place. As a workaround, incremental tar files can be used, which are separate tar files that consist of files updated after the time the original or subsequent incremental updates were created.

Creating Tar Files

This example creates a tar file (*dir1*.tar) of the directory *dir1* on your local system directly into your Lou home directory, along with a corresponding table of contents (*dir1*.tar.toc):

your_local_system% sup shiftc --create-tar --index-tar *dir1* lfe:*dir1*.tar

Note:? If the *dir1* directory is over 1 TB, it will be split into multiple tar files prefixed with *dir1*.

Creating Incremental Tar Files

The example in this section shows you how to create incremental tar files from your local system to your Lou home directory.

Create an incremental tar file (*dir1*-2020.tar) of all files modified on or after January 1st, 2020 in the directory *dir1* on your local system directly to your Lou home directory, along with a corresponding table of contents (*dir1*-2020.tar.toc):

your_local_system% sup shiftc --create-tar --index-tar --newer="Jan 1 2020" *dir1* lfe:*dir1*-2020.tar

Extracting Tar Files

These examples show you how to extract tar files from your local system to your Lou home directory.

- Extract the *dir1*.tar file from your local system to your Lou home directory:

  your_local_system% sup shiftc --extract-tar lfe:*dir1*.tar .

- If *dir1* was over 1 TB and therefore split into multiple tar files:

  your_local_system% sup shiftc --extract-tar lfe:'*dir1.**tar' .

  Note: As shown in this example, quotation marks are required when using wildcards (e.g., *) with sources on a remote host.

- Extract the files *1g.20* through *1g.29* from *dir1*.tar from your local system to your Lou home directory:

  your_local_system% sup shiftc --extract-tar --include='1g\.2[0-9]' lfe:*dir1*.tar .

  Note: As shown in this example, shiftc uses Perl-style regular expressions for some options.

Local Transfers

Checking File Integrity

It is a good practice to confirm whether your files are complete and accurate before you transfer the files to or from NAS, and again after the transfer is complete.

The easiest way to verify the integrity of file transfers is to use the NAS-developed Shift tool for the transfer, with the --verify option enabled. As part of the transfer, Shift will automatically checksum the data at both the source and destination to detect corruption. If corruption is detected, partial file transfers/checksums will be performed until the corruption is rectified.

For example:

pfe*21*% shiftc --verify $HOME/*filename* /nobackupp*X*/*username*
lou% shiftc --verify /nobackupp*X*/*username/filename* $HOME
your_localhost% sup shiftc --verify *filename* pfe:

In addition to Shift, there are several algorithms and programs you can use to compute a checksum. If the results of the pre-transfer checksum match the results obtained after the transfer, you can be reasonably certain that the data in the transferred files is not corrupted. If data *is* corrupted during a transfer, a good checksum algorithm will yield different results before and after the transfer, with high probability.

The following checksum programs are available on HECC systems:

sum
: Computes a checksum using the BSD sum or System V sum algorithm; also counts the number of blocks (1 KB-block or 512 B-block) in a file

cksum
: Computes a cyclic redundancy check (CRC) checksum; also counts the number of bytes in a file

md5sum
: Computes a 128-bit MD5 checksum, which is represented by a 32-character hexadecimal number

msum
: High performance multi-threaded checksum utility, developed at NAS. By default, computes 128-bit MD5 checksums, but you can compute other types using --hash-type. Note that for full compatibility with md5sum output, you must use the --split-size=0 option, which will also decrease performance on large files. For more information, see **man msum** on any HECC front-end system.

For example:

% ls -l *filename*
-rw------- 1 *username group_id* 67358 Nov 15 11:49 *filename*

% sum *filename*
50063 66

% cksum *filename*
269056887 67358 *filename*

% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

% msum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

To check the integrity of an existing tar file against a directory:

% tar -tf *dir_name*.tar | sort > sums.*dir_name*.tar
% find *dir_name* | xargs md5sum | sort>sums.*dir_name*.dir
% diff sums.*dir_name*.tar sums.*dir_name*.dir

The md5sum utility is more reliable than the sum or cksum commands for detecting accidental file corruption, as the chances of accidentally having two files with identical MD5 checksums are extremely small. The utility is installed by default in most Unix, Linux, and Unix-like operating systems. We recommend that you compute the md5sum of a file before and after the transfer.

The following example shows that the file *filename* is complete and accurate after the transfer, based on its md5sum.

pfe*21*% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

pfe*21*% scp *filename local_username@your_localhost*:

your_localhost%md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

See the **sum, cksum, md5sum, and msum man pages** for more information on these commands.

For large file transfers within the NAS enclave, use the commands shiftc, mcp, or cxfscp. The slower cp command also can be used.

The following file transfer commands can be used when both the source and destination locations are accessible on the same host where the command is issued.

Shift Command

Shift (shiftc) is a NAS-developed tool for performing automated local and remote file transfers. Shift utilizes a variety of underlying file transports to achieve maximum performance for files of any size on any file system.

Where is it installed at NAS?

shiftc is installed on the Pleiades and Lou front-end systems (PFEs and LFEs).

When to use it?

The command shiftc can be used as a drop-in replacement for cp at any time on any system on which it is installed.

Examples

lfe*5*% shiftc /nobackup/*username/filename* $HOME
pfe*27*% shiftc $HOME/*filename* /nobackup/*username*

Performance

shiftc utilizes mcp, when available, so it can be up to 10 times faster than cp for large files (2+ GB) and can achieve up to 1.8 GB/sec on a single host. Using the --hosts option, it has been measured to achieve up to 5.0 GB/sec on 8 hosts.

For more information, see Shift File Transfer Overview.

cxfscp

cxfscp is a program for quickly copying large files to and from a CXFS filesystem, such as the Lou front-end (LFE) home file system. It can be significantly faster than cp on CXFS filesystems since it uses multiple threads and large direct I/Os to fully utilize the bandwidth to the storage hardware.

For files less than 64 kilobytes in size, which will not benefit from large direct I/Os, cxfscp will use a separate thread for copying these files using buffered I/O similar to cp.

Where is it installed at NAS?

cxfscp is installed on the LFEs.

When to use it?

The Pleiades Lustre filesystems (/nobackupp*X*) are mounted on lfe[*5-8*]. The command cxfscp can be issued on either of these hosts to copy large files between the LFE's CXFS home file system and Pleiades's /nobackup. This is an easy way to transfer files between Lou and Pleiades without the need for scp or scp.

Examples

lfe*5*% cxfscp -rp --bi /nobackupp4/*username*/*testdir*_a /u/*username*/tests
lfe*5*% cxfscp -p --bo /u/*username*/data* /nobackupp4/*username*/*data_dir*

Performance

Some benchmarks done by NAS staff show that cxfscp is typically 4-7 times faster than cp for large files (2+ GB) and can achieve up to 650 MB/sec.

For more information, read **man cxfscp**.

cp

cp is a Unix command for copying files between two locations (for example, two different directories of the same filesystem or two different filesystems such as NFS, CXFS or Lustre).

Where is it installed at NAS?

cp is available on all NAS systems except the secure front ends (SFEs).

Use the -p option to preserve the timestamp on the file.

Examples

pfe*21*% cp -p $HOME/*filename* $HOME/*newdir/filename2*
pfe*21*% cp -p $HOME/*filename* /nobackup/*username*

The NAS-developed Shift tool can copy files locally on NAS enclave hosts, transfer files between hosts inside the NAS enclave, and transfer files between the NAS enclave and remote hosts. You can also use Shift to check the status of transfers at any time, receive email notification of completion, errors, and warnings, and restart interrupted transfers or transfers with errors.

All functionality is accessed through the Shift client, which is invoked via the `shiftc` command. The syntax for `shiftc` is similar to the syntax for the cp and scp commands.

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Advanced Features

Shift includes the following advanced features:

- Automatic parallelization of transfers
- Local and remote tar creation and extraction
- Synchronization based on modification times and checksums (similar to `rsync`)
- Automatic file integrity verification and correction
- Ability to restart transfers
- Automatic retrieval of files from tape storage (DMF-managed Lou filesystems)
- Ability to check status of current transfers

How to Use Shift

See the following articles for detailed information about how to use Shift:

- Using Shift for Local Transfers and Tar Operations
- Using Shift for Transfers and Tar Operations Between Two NAS Hosts
- Using Shift for Remote Transfers and Tar Operations
- Checking Shift Transfer Status and Restarting Transfers
- Shift Command Options

Additional Resources

You can also see presentation slides for three HECC training webinars that demonstrate how to use the tool:

- Simplifying and Optimizing Your Data Transfers (PDF)
- Advanced Features of the Shift Automated File Transfer Tool (PDF)
- Simple Automated File Transfers Using SUP and Shift (PDF)

Recordings of each presentation are also available in the Past Webinars Archive.

Note: Some hostnames and options may have changed since the webinars were presented.

Remote Transfers

Use the file transfer commands scp or shiftc when the source and destination are located on different hosts—either on two different NAS high-end computing hosts, or on a NAS host and a remote host such as your local desktop system.

The basic syntax is:

copy-command [options]...source...destination

scp Command

The Secure Copy Protocol (scp) command, based on the Secure Shell Protocol (SSH), is a means of securely transferring files between a local and a remote host. Both your authentication information (such as password or passcode) and your data are encrypted.

The most widely used scp program is from OpenSSH.

Where is scp installed at NAS?

A copy of scp from OpenSSH without the patch is available on the Pleiades front-end systems (PFEs), Lou, and the secure front-end systems (SFEs).

Do you need it installed on your local host?

If you have a version of SSH installed on your local host, scp is most likely already installed there.

When to use it?

Typically, scp is used to transfer small files within NAS (<< 5 GB) or offsite (<< 1 GB) that take a reasonable amount of time to complete.

Examples

In these examples, "outbound" means the command is initiated on a NAS host such as Pleiades or Lou, whether the file is being pushed or pulled. "Inbound" means the command is initiated on your local host.

Note: Omit *local_username@* and *nas_username@* in the examples below if your local username and NAS username are identical. These examples assume you already know how to log into the NAS enclave.

For outbound transfer:

lou% scp *local_username@your_localhost.domain:file1 ./file2*

For inbound transfer if SSH passthrough has been set up correctly:

your_localhost% scp *file1 nas_username*@lou.nas.nasa.gov:*file2*

For inbound transfer if you have not set up SSH passthrough:

your_localhost% scp -oProxyCommand='ssh *nas_username*@sfeX.nas.nasa.gov
ssh-proxy %h'*file1 nas_username*@lou.nas.nasa.gov:*file2*

where sfeX is sfe[6-8].

Note: Due to formatting issues, the command line in the above example is shown on two lines. It should be entered as one line.

Performance

Within the NAS secure enclave, depending on source and destination hosts and other factors, the performance range will be 40-100 MB/sec.

If your data will compress well, consider enabling compression by adding -C to your scp command line.

We recommend using OpenSSH 5.0 or a newer version.

Shift Command (shiftc)

Shift (shiftc) is a NAS-developed tool for performing automated local and remote file transfers. Shift utilizes a variety of underlying file transports to achieve maximum performance for files of any size on any file system.

Where is it installed at NAS?

Shift is installed on Lou and on the PFEs.

Do you need it installed on your local host?

For transfers between your local host and NAS systems, you must install the SUP client as discussed in Shift File Transfer Overview.

When to use it?

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Shift can be used as a drop-in replacement for scp between any enclave systems. For transfers between your local host and NAS systems, the transfer must be initiated from your local host with shiftc invoked via the SUP client (that is, using the command sup shiftc). If an encrypted transfer is required, use the shiftc --secure option.

Example

pfe27% shiftc /nobackupp2/*username/filename* lou:
your_localhost% sup shiftc pfe:*filename* .

Performance

Shift uses the highest performing file transport that is available on both sides of the transfer, and is optimal for the sizes of the files being transferred.

For more information, see File Transfer: Overview and Shift File Transfer Overview.

Checking File Integrity

It is a good practice to confirm whether your files are complete and accurate before you transfer the files to or from NAS, and again after the transfer is complete.

The easiest way to verify the integrity of file transfers is to use the NAS-developed Shift tool for the transfer, with the --verify option enabled. As part of the transfer, Shift will automatically checksum the data at both the source and destination to detect corruption. If corruption is detected, partial file transfers/checksums will be performed until the corruption is rectified.

For example:

pfe*21*% shiftc --verify $HOME/*filename* /nobackupp*X*/*username*
lou% shiftc --verify /nobackupp*X*/*username/filename* $HOME
your_localhost% sup shiftc --verify *filename* pfe:

In addition to Shift, there are several algorithms and programs you can use to compute a checksum. If the results of the pre-transfer checksum match the results obtained after the transfer, you can be reasonably certain that the data in the transferred files is not corrupted. If data *is* corrupted during a transfer, a good checksum algorithm will yield different results before and after the transfer, with high probability.

The following checksum programs are available on HECC systems:

sum
: Computes a checksum using the BSD sum or System V sum algorithm; also counts the number of blocks (1 KB-block or 512 B-block) in a file

cksum
: Computes a cyclic redundancy check (CRC) checksum; also counts the number of bytes in a file

md5sum
: Computes a 128-bit MD5 checksum, which is represented by a 32-character hexadecimal number

msum
: High performance multi-threaded checksum utility, developed at NAS. By default, computes 128-bit MD5 checksums, but you can compute other types using --hash-type. Note that for full compatibility with md5sum output, you must use the --split-size=0 option, which will also decrease performance on large files. For more information, see **man msum** on any HECC front-end system.

For example:

% ls -l *filename*
-rw------- 1 *username group_id* 67358 Nov 15 11:49 *filename*

% sum *filename*
50063 66

% cksum *filename*
269056887 67358 *filename*

% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

% msum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

To check the integrity of an existing tar file against a directory:

% tar -tf *dir_name*.tar | sort > sums.*dir_name*.tar
% find *dir_name* | xargs md5sum | sort>sums.*dir_name*.dir
% diff sums.*dir_name*.tar sums.*dir_name*.dir

The md5sum utility is more reliable than the sum or cksum commands for detecting accidental file corruption, as the chances of accidentally having two files with identical MD5 checksums are extremely small. The utility is installed by default in most Unix, Linux, and Unix-like operating systems. We recommend that you compute the md5sum of a file before and after the transfer.

The following example shows that the file *filename* is complete and accurate after the transfer, based on its md5sum.

pfe*21*% md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

pfe*21*% scp *filename local_username@your_localhost*:

your_localhost%md5sum *filename*
cfe0fc62607e9dc6ea0c231982316b75 *filename*

See the **sum, cksum, md5sum, and msum man pages** for more information on these commands.

Encryption helps protect your files during inter-host file transfers that use protocols that are not already encrypted—for example, when using ftp or when using shiftc without the --secure option. We recommend using the GNU Privacy Guard (GPG), an Open Source OpenPGP-compatible encryption system.

GPG has been installed on Pleiades, Endeavour, and Lou in the /usr/bin/gpg directory. If you do not have GPG installed on the system(s) that you would like to use for transferring files, please see the GPG website.

Choosing What Cipher to Use

We recommend using the cipher AES256, which uses a 256-bit Advanced Encryption Standard (AES) key to encrypt the data. Information on AES can be found at the National Institute of Standards and Technology's Computer Security Resource Center.

You can set your cipher in one of the following ways:

- Add --cipher-algo AES256 to your ~/.gnupg/gpg.conf file.
- Add --cipher-algo AES256 in the command line to override the default cipher, CAST5.

Examples

If you choose not to add the cipher-algo AES256 to your gpg.conf file, you can add --cipher-algo AES256 on any of these simple example command lines to override the default cipher, CAST5.

Creating an Encrypted File

Both commands below are identical. They encrypt the test.out file and produce the encrypted version in the test.gpg file:

```
% gpg --output test.gpg --symmetric test.out
```

```
% gpg -o test.gpg -c test.out
```

You will be prompted for a passphrase, which will be used later to decrypt the file.

Decrypting a File

The following command decrypts the test.gpg file and produces the test.out file:

```
% gpg --output test.out -d test.gpg
```

You will be prompted for the passphrase that you used to encrypt the file. If you don't use the --output option, the command output goes to STDOUT. If you don't use any flags, it will decrypt to a file without the .gpg suffix. For example, using the following command line would result in the decrypted data in a file named "test":

```
% gpg test.gpg
```

Selecting a Passphrase

Your passphrase should have sufficient information entropy. We suggest that you include five words of 5-10 letters in size, chosen at random, with spaces, special characters, and/or numbers embedded into the words.

You need to be able to recall the passphrase that was used to encrypt the file.

Factors that Affect Encrypt/Decrypt Speed on NAS Filesystems

We do not recommend using the --armour option for encrypting files that will be transferred to/from NAS systems. This option is mainly intended for sending binary data through email, not via transfer commands such as ftp or shiftc with the -secure option. The file size tends to be about 33% bigger than without this option, and encrypting the data takes about 10-15% longer.

The level of compression used when encrypting/decrypting affects the time required to complete the operation. There are three options for the compression algorithm: none, zip, and zlib.

- --compress-algo none or --compress-algo 0
- --compress-algo zip or --compress-algo 1
- --compress-algo zlib or --compress-algo 2

For example:

```
% gpg --output test.gpg --compress-algo zlib --symmetric test.out
```

If your data is not compressible, --compress-algo 0 (none) gives you a performance increase of about 50% compared to --compress-algo

1 or --compress-algo 2.

If your data is highly compressible, choosing the zlib or zip option will not only increase the speed by 20-50%, it will also reduce the file size by up to 20x. For example, in one test on a NAS system, a 517 megabyte (MB) highly compressible file was compressed to 30 MB.

The zlib option is not compatible with PGP 6.*x*, but neither is the cipher algorithm AES256. Using the zlib option is about 10% faster than using the zip option on a NAS system, and zlib compresses about 10% better than zip.

Random Benchmark Data

We tested the encryption/decryption speed of three different files (1 MB, 150 MB, and 517 MB) on NAS systems. The file used for the 1 MB test was an RPM file, presumably already compressed, since the resulting file sizes for the none/zip/zlib options were within 1% of each other. The 150 MB file was an ISO file, also assumed to be a compressed binary file for the same reasons. The 517 MB file was a text file. These runs were performed on a CXFS filesystem when many other users' jobs were running. The performance reported here is for reference only, and not the best or worst performance you can expect.

Using AES256 as the Cipher Algorithm			
	1 MB File	**150 MB File**	**517 MB File**
with --armour	~5.5 secs to encrypt	~40 secs to encrypt	
without --armour	~4 secs to encrypt	~35 secs to encrypt	
without --armour, zlib compression		~33 secs to encrypt; ~28 secs to decrypt to file	~33 secs, resultant file size ~30 MB; ~34 secs to decrypt to file
without --armour, zip compression		~36 secs to encrypt; ~31 secs to decrypt to file	~38 secs, resultant file size ~33 MB; ~34 secs to decrypt to file
without --armour, no compression		~19 secs to encrypt; ~25 secs to decrypt to file	~49 secs, resultant file size ~517 MB; ~75 secs to decrypt to file

Shift Transfer Tool Overview

The NAS-developed Shift tool can copy files locally on NAS enclave hosts, transfer files between hosts inside the NAS enclave, and transfer files between the NAS enclave and remote hosts. You can also use Shift to check the status of transfers at any time, receive email notification of completion, errors, and warnings, and restart interrupted transfers or transfers with errors.

All functionality is accessed through the Shift client, which is invoked via the `shiftc` command. The syntax for `shiftc` is similar to the syntax for the cp and scp commands.

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Advanced Features

Shift includes the following advanced features:

- Automatic parallelization of transfers
- Local and remote tar creation and extraction
- Synchronization based on modification times and checksums (similar to `rsync`)
- Automatic file integrity verification and correction
- Ability to restart transfers
- Automatic retrieval of files from tape storage (DMF-managed Lou filesystems)
- Ability to check status of current transfers

How to Use Shift

See the following articles for detailed information about how to use Shift:

- Using Shift for Local Transfers and Tar Operations
- Using Shift for Transfers and Tar Operations Between Two NAS Hosts
- Using Shift for Remote Transfers and Tar Operations
- Checking Shift Transfer Status and Restarting Transfers
- Shift Command Options

Additional Resources

You can also see presentation slides for three HECC training webinars that demonstrate how to use the tool:

- Simplifying and Optimizing Your Data Transfers (PDF)
- Advanced Features of the Shift Automated File Transfer Tool (PDF)
- Simple Automated File Transfers Using SUP and Shift (PDF)

Recordings of each presentation are also available in the Past Webinars Archive.

Note: Some hostnames and options may have changed since the webinars were presented.

Using SUP for Remote Transfers

UPDATE: As of June 2, 2023, you must ensure that your SUP client has been updated with the latest version 8.13 (available below). You must do one of the following to replace their SUP client manually:

1. Follow the installation instructions for the SUP client.
   or
2. Run the following command directly on top of the old client, or to a new file:
   ssh sup.nas.nasa.gov mesh-update --file=mc>sup
   If downloading the client to a new location, use chmod 700 sup to make the client executable, then move it to a location in your $PATH.

The Secure Unattended Proxy (SUP) allows you to perform remote operations on specific hosts within the NAS enclave (currently, the Pleiades and Lou front-end systems [PFEs and LFEs]) *without* needing to use your RSA SecurID token at the time of the operation.

To accomplish this, you must first obtain special "SUP keys" using RSA SecurID authentication, which you can then use to perform operations from unattended jobs and/or scripts. Each SUP key is valid for a period of *one week* from the time it is generated. You may have multiple SUP keys at the same time, which will expire asynchronously.

SUP keys are currently allowed to call scp, sftp, qstat, rsync, shiftc, ssh-balance, and test. In the future, other operations may be available via the SUP.

Note: The shiftc command is built into the sup command; therefore, when you download the SUP client, you will also get shiftc.

SUP Usage Summary

The steps in this section demonstrate how to quickly get up and running with the SUP. Each step is explained in more detail in subsequent sections.

In these steps, *host* refers to a PFE or LFE; the command lines shown are examples.

1. Download and install the SUP client in your personal bin directory (one time).

   your_localhost% wget -O sup https://www.nas.nasa.gov/hecc/support/kb/file/9
   your_localhost% chmod 700 sup
   your_localhost% mv sup ~/bin

2. Authorize host for SUP operations (one time per host):

   pfe% touch ~/.meshrc

3. Authorize directories for write operations (one or more times per host):

   pfe% echo /nobackupp2 >> ~/.meshrc

4. Execute a command (each time)

   your_localhost% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

5. Examine expected output (as needed)
6. Troubleshoot problems (as needed)

SUP Client

The SUP client performs all the steps needed to execute commands through the SUP as if the SUP itself did not exist. Commands that are allowed to pass through the SUP can be executed as if the remote host were directly connected by simply prepending the client command sup, as shown in example 4, above. Besides executing remote commands, the client also includes an operating-system-independent virtual filesystem that allows files across all SUP-connected resources to be accessed using standard filesystem commands.

Requirements

The SUP requires OpenSSH version 5.4 (or later) and Perl version 5.8.5 (or later) and has been tested successfully on Linux, OS X, and Windows under Cygwin. These requirements are typically installed by default on Linux and OS X, but may need to be manually installed on Cygwin through the package manager as the openssh and perl packages. For general Cygwin installation instructions, see Installing Cygwin (PDF).

Installation

Complete the following steps to install the client:

1. Download the client and save it to a file called *sup*
2. Make the client executable by using chmod 700 sup
3. Move the client to a location in your $PATH

SSH Configuration

If your local username differs from your NAS username, it is recommended that you add the following to your~/.ssh/config file, where *nas_username* is replaced with your NAS username:

```
Host sup.nas.nasa.gov sup-key.nas.nasa.gov
    User nas_username
```

Note: If you are using a config file based on the NAS config template, you do not have to do this step.

Alternatively, the client's -u option can be used as described in the next section. If your local username is the same as your NAS username, no additional configuration or command-line options are required.

SUP Command-Line Options

-b
: Disable user interaction; for use within scripts. Note that the client will fail if any interaction is required--normally only needed when your SUP key has expired or is otherwise unavailable.

-u *username*
: Specify NAS username. Note that this option is required if your local username differs from your NAS username and you have not modified your SSH configuration appropriately.

-v
: Enable verbose output for debugging purposes.

SUP Authorizations

The basic set of operations that may be performed using the SUP is specified by the administrator. To protect accounts from malicious use of SUP keys, users must grant execute and write permissions to SUP operations on each target system.

Authorizing Execute Operations

By default, even SUP operations permitted by site policy are not allowed to execute on a given host. To enable SUP operations to a given host (PFE or LFE), the file ~/.meshrc must exist on that host, which can be created by invoking the following:

```
% touch ~/.meshrc
```

Note that the PFEs share their home filesystems, so this must only be done on one of these nodes. Other systems must be authorized separately. Once this file exists on a host, all operations permitted by site policy are allowed to execute on that host.

Authorizing Write Operations

By default, SUP operations are not allowed to write to the filesystem on a given host. To enable writes to a given directory on a given host, that directory must be added (on a separate line) to the ~/.meshrc file on that host. For example, the following lines in ~/.meshrc indicate that writes should be permitted to /nobackupp2 and your home directory.

```
/nobackupp2
/u/username
```

Each directory is the root of allowed writes, so this configuration would allow writes to all files and directories rooted at /nobackupp2 and your home directory (for example, /nobackupp2/*some/dir*).

Note that the root directory cannot be authorized. Also note that dot files (that is,~/.*) in your home directory are never writable regardless of the contents of ~/.meshrc.

Executing Commands Through SUP

Usage examples of each command that may be executed through the SUP are given below.

Note: SUP commands must be authorized for execution on each target host, and that transfers to a given host must beauthorized for writes. Before a given operation is performed, the client may ask for certain information, including the existing or new passphrase for ~/.ssh/id_rsa, the password + passcode for sup.nas.nasa.gov, and/or the password + passcode for sup-key.nas.nasa.gov.

For more detailed information on each command, see their corresponding **man pages**.

File Transfer Commands

rsync

your_localhost% sup rsync *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

If you intend to transfer files to your home directory, note that even if your home directory has beenauthorized for writes, rsync transfers to your home directory will fail unless the --inplace option is specified. This is because rsync uses temporary files starting with "." during transfers, which cannot be written in your home directory. You can avoid this problem by specifying --inplace as shown

in the following example:

your_localhost% sup rsync --inplace *foobar* pfe*21*:

scp

your_localhost% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

sftp

your_localhost% sup sftp pfe*21*

shiftc

your_localhost% sup shiftc *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

For more information, see Using Shift for Remote Transfers and Tar Operations.

File Monitoring Command

test

your_localhost% sup ssh pfe*21* test -f /u/*username*/c_*foobar*

Job Monitoring Command

qstat

your_localhost% sup ssh pfe*21* qstat @pbspl1

ssh-balance and bbFTP

ssh-balance

your_localhost% sup ssh-balance -l pfe

Note that this command allows the Pleiades load balancer to be used with bbftp:

your_localhost% sup bbftp -e "put *foobar* /u/*username*/*c_foobar*" `sup ssh-balance -l pfe`

SUP Expected Output

The following sequence shows the expected output for the command:

your_localhost% sup scp *foobar* pfe*21*:/nobackupp2/*username*/c_*foobar*

for a user who has never used the SUP before.

The conditions under which each sub-sequence will be seen are indicated next to each header. Most of the items will only be seen once or during key generation. A second invocation will only show the command output portion.

1. **Identity initialization** (seen once per identity)

   Initializing identity on sup-key.nas.nasa.gov (provide login information)
   (username@sup-key.nas.nasa.gov) Password:
   (username@sup-key.nas.nasa.gov) Enter PASSCODE:
   key SHA256:p3VqOcfV46fJwhOpYs/2h2sztRkNZ3nFu6VQbl2pgfI uploaded successfully

2. **SUP key generation** (seen when no valid SUP keys available)

   Generating key on sup.nas.nasa.gov (provide login information)
   (username@sup.nas.nasa.gov) Password:
   (username@sup.nas.nasa.gov) Enter PASSCODE:

3. **Client upgrade** (seen during key generation when new client available)

   A newer version of the client is available (8.04 vs. 8.1)
   ...do you wish to replace the current version? (y/n) **y**

4. **Command output** (always seen)

   foobar 100% 5 0.0KB/s 00:00

SUP Troubleshooting

The following error messages may be encountered during your SUP client usage. Note that the-v option can be given to the SUP client to output additional debugging information.

- "WARNING: Your password has expired"

This message indicates that your current password has expired and must be changed. To change your password, you must log in to an LDAP host (for example, Lou) through an SFE and change your LDAP password. This change will be automatically propagated to the SUP within a few minutes.

- "Permission denied (~/.meshrc not found)"

  This message indicates that you have not created a .meshrc file in your home directory on the target host. SUP commands must be authorized for execution on each target host.

- "Permission denied (unauthorized command)"

  This message indicates that you have attempted an operation that is not currently authorized by the SUP. Check that the command line is valid and that the attempted command is one of the authorized commands. Certain options to authorized commands may also be disallowed, but these should never be needed in standard usage scenarios.

- "Permission denied during file access" (various forms)

  These messages indicates that you attempted to read or write a file for which such access is not allowed. The most common cause is forgetting to authorize directories for writes. Reads and writes of ~/.* are never permitted.

- "Permission denied (publickey)"

  This message indicates that you may not have proper permissions on your ~/.ssh and/or home directory on the target host. Check to make sure that ~/.ssh is not readable/writable by other users/groups and that your home directory is not writable by other users/groups.

The SUP virtual file system (VFS) capability can be used to issue file-related commands to all SUP-connected hosts from the command line of your local host.

Accessing Files Across Multiple Hosts

The SUP client includes a virtual file system capability that allows files across all SUP-connected resources to be accessed using standard file system commands. For example, once you have activated VFS as described below, the command:

% ls pfe*21*:/tmp

from your local host would list the files in /tmp on pfe*21*. The command:

% cp *filename* pfe*21*:/tmp

from your local host would copy the file *filename* from your current directory on your local host to /tmp on pfe*21*.

The set of supported commands includes cat, cd, chgrp, chmod, chown, cmp, cp, df, diff, du, file, grep, head, less, ln, ls, mkdir, more, mv, pwd, rm, rmdir, tail, tee, test, touch, and wc.

Note that this functionality is not a true file system since only these commands are supported and only when used from within a shell. Unlike more general approaches such as FUSE, however, the SUP capability is completely portable and can be enabled with no additional privileges or software.

Commands through the VFS functionality can act on any combination of local and remote files, where remote files are prefixed with *hostname*:. For example, the command:

% cat pfe*21*:/tmp/*rfile* ~/*lfile*

would print the file *rfile* in /tmp on pfe*21* as well as the file *lfile* in the user's home directory on the local host to the terminal. Any number of hosts can be included in any command. For example, the command:

% diff lfe2:/tmp/*lfe_file* pfe*21*:/tmp/*pfe_file*

would show the differences between the file *lfe_file* in /tmp on lfe2 and the file *pfe_file* in /tmp on pfe*21*. The client determines if any remote access is needed based on the path(s) given. If not, it will execute the command locally as given as rapidly as possible. Fully local commands also support all options with the exception of options of the form -f *value* (that is, single-dash options that take values).

VFS Activation

Requirements

Currently, SUP VFS functionality is only supported for bash, but csh support is planned for the future. This functionality requires Perl version 5.8.5 (note that this is more recent than version 5.6.1 required by the basic client functionality). It also requires the standard Unix utilities cat, column, false, sort, and true; and has been tested successfully on Linux, OS X, and Windows under Cygwin. Note that users of Windows under Cygwin may need to install the coreutils and util-linux packages to obtain these utilities.

Activation and Deactivation on Your Local Host

Complete these steps.

1. Install the SUP client if you have not already done so.
2. Activate VFS functionality in a bash shell.
    - For an interactive bash shell, run:

      ```
      eval `sup -s bash`
      ```

    - For a non-interactive bash shell, begin the script with the following:

      ```
      #!/bin/bash
      shopt -s expand_aliases
      eval `sup -s bash`
      ```

    The instructions will load aliases and functions that will intercept specific commands and replace them with commands that will perform the requested actions through the SUP client.

To deactivate VFS functionality in a bash shell or bash script at any time, run:

% eval `sup -r bash`

Command-Line Options

The behavior of the virtual file system can be modified using various options at the time it is activated.

-ocmd=opts
: Specify default options for a given command since the VFS functionality overrides any existing aliases for its supported set of commands.

-t *transport*
: Change the file transport from its sftp default to transport. Currently, the only additional transport available is bbftp. Note that using bbftp as the transport may slow down certain operations on small files as bbftp has higher startup overhead.

-u *user*
: Specify NAS user name. Note that this option is *required* if your local user name differs from your NAS user name.

For example, the following invocation activates the client virtual file system using bbftp as the transport mechanism, *nasuser* as the user and adds colorization of local file listings using the Linux ls --color=always option.

```
% eval `sup -s bash -t bbftp -u nasuser -ols=--color=always`
```

VFS Caveats

The VFS functionality is still somewhat experimental. In general, it works for the most common usage scenarios with some caveats. In particular:

- "Whole file" commands (that is, commands that must process the entire file), including cat, cmp, diff, grep, wc (and currently more/less due to implementation) retrieve files first before processing for efficiency. Thus, these commands should not be executed on very large files.
- There is a conflict between commands that take piped input and the custom globbing of the client, thus these commands have portions of globbing support disabled. These commands are grep, head, less, more, tail, tee, and wc. In these cases, globbing will work for absolute prefixes, but not relative. For example, grep *filename* pfe*21:*/tmp/* will work, but cd pfe*21:*/tmp; grep *filename* * will not.
- Redirection to/from remote files doesn't work. The same effect can be achieved using cat and tee (for example, grep localhost < pfe*21:*/etc/hosts > pfe*21:*/tmp/a could be done with cat pfe*21:*/etc/hosts | grep localhost | tee pfe*21:*/tmp/a > /dev/null). Redirection still works normally for local files.
- The first time a command is run involving a particular host, a SFTP connection is created to that host. When runningps, it may appear as if a zombie client process is running.

VFS Commands

Currently supported commands and their currently supported options are below. Unsupported options will simply be ignored except where noted. All commands are still subject to SUP authorizations, thus something that cannot be executed or written normally through the SUP cannot be executed or written through this functionality either.

- cat (no options)
- cd (no options)
  Note that when changing to remote directories, cd only changes $PWD, so to make changes visible the working directory (that is, \w in bash) must be in your prompt. For example, the following prompt:

  ```
  export PS1="\h[\w]> "
  ```

  would display the current host name followed by the current working directory.
- chgrp (no options)
  Groups may be specified either by number or by name. Names will be resolved on the remote host.
- chmod (no options)
  Modes must be specified numerically (for example, 0700). Symbolic modes, such as a+rX, are not currently supported.
- chown (no options)
  Users and groups may be specified either by number or by name. Names will be resolved on the remote host.
- cmp (all options)
- cp [-r]
  Note that copies between two remote hosts transfer files to the local host first since the SUP does not allow third party transfers. Thus, very large file transfers between remote systems should be achieved using an alternate approach.
- df [-i]
  Note that 1024-byte blocks are used.
- diff (all options)
- du [-a] [-b] [-s]
  Note that 1024-byte blocks are used.
- file (all options)
- grep (all options)
- head [-*number*]
  Note that head does not support the form -n *number*, so, for example, to display the first 5 lines of a file, use -5 and not -n 5.
- less (all options)
- ln [-s]
  Note that hard links are not supported. Links from remote files to local files (for example, ln -s pfe*21:*/*filename* /*filename*) will be dereferenced during certain operations (for example, cat /*filename* will cat pfe*21:*/*filename*).
- ls [-1] [-d] [-l]

For efficiency purposes, ls behaves slightly differently for remote commands than for local commands. In particular ls -l will not show links by default and will show what is actually linked instead of the link itself. Link details can be obtained using the -d option (for example, ls -ld *).

Also for efficiency, ls processes remote files before local files, so output ordering may be changed when remote and local files are interleaved on the ls command line. For example, ls /*file1* pfe*21*: /*file2* would show pfe*21*: first, then /*file1*, then /*file2*.

- mkdir (no options)
- more (all options)
- mv (no options)
- pwd (no options)
- rm [-r]
- rmdir (no options)
- tail [-*number*]
  Note that tail does not support the form -n *number*, so, for example, to display the last 5 lines of a file, use -5 and not -n 5.
- tee [-a]
- test [-b] [-c] [-d] [-e] [-f] [-g] [-h] [-k] [-L] [-p] [-r] [-s] [-S] [-u] [-w]
  Note that compound and string tests are not supported. Compound and string tests can be achieved using multiple test commands separated by shell compound operators. For example, instead of

  % test -f pfe*21*:/*filename* -a "*abc*" != "*123*"

  do

  % test -f pfe*21*:/*filename* && test "*abc*" != "*123*"

  Alternatively, the actual test command can be executed through the SUP:

  % sup ssh pfe*21* test -f /*filename* -a "*abc*" != "*123*"

- touch (no options)
- wc (all options)

Examples

Inbound file transfers through the secure front ends (SFEs) require RSA SecurID token authentication. Files cannot be transferred directly to the SFEs, so transfers must be done using either SSH passthrough or using scp with the -oProxyCommand option.

Note: While this article covers file transfers through the SFEs only, you can also transfer files through the Secure Unattended Proxy.

To simplify the instructions, the approaches are described in terms of transfers to or from one of the Pleiades front ends (PFEs), such as pfe21, but they also apply to any of the other systems that are in the secure enclave—such as other PFEs, or the Lou front ends (LFEs).

For some of the methods described, two commands are provided. The first command (a) is used if you have identical usernames on your local system and on the NAS systems, or if the usernames are different but you have set up your local ~/.ssh/config file to include the NAS username. To learn how to set this up, download the ~/.ssh/config template. The second command (b) is used if your usernames are different and you do not include the NAS username in your local ~/.ssh/config file.

File Transfers Using scp with -oProxyCommand

The scp command is not recommended for files over 1 GB; see Remote File Transfer Commands for a comparison of commands. If you have not set up SSH passthrough, you must use scp with the -oProxyCommand option for inbound file transfers through the SFEs.

Note: Because of a formatting issue, the commands in this section are broken into two lines. Each should be on only one line.

Using scp with the -oProxyCommand option to *push* files *out* of your local system:

(a) *your_local_system*% scp -oProxyCommand='ssh sfe6.nas.nasa.gov
ssh-proxy %h' *filename* pfe21.nas.nasa.gov:

or

(b) *your_local_system*% scp -oProxyCommand='ssh *nas_username*@sfe6.nas.nasa.gov
ssh-proxy %h' *filename nas_username*@pfe21.nas.nasa.gov:

Using scp with the -oProxyCommand option to *pull* files *into* your local system:

(a) *your_local_system*% scp -oProxyCommand='ssh sfe6.nas.nasa.gov
ssh-proxy %h' pfe21.nas.nasa.gov:*filename* .

or

(b) *your_local_system*% scp -oProxyCommand='ssh *nas_username*@sfe6.nas.nasa.gov
ssh-proxy %h' *nas_username*@pfe21.nas.nasa.gov:*filename* .

File Transfers Using SSH Passthrough

If you have set up SSH passthrough correctly, you can use scp or shiftc to transfer files between your local system and a NAS host. You can transfer files directly into the NAS host and avoid the need to double-authenticate. The passage through the SFEs is transparent.

Using scp

Using scp to *push* files *out* of your local system:

(a) *your_local_system*% scp *filename* pfe*21*.nas.nasa.gov:

or

(b) *your_local_system*% scp *filename nas_username*@pfe*21*.nas.nasa.gov:

Using scp to *pull* files *into* your local system:

(a) *your_local_system*% scp pfe*21*.nas.nasa.gov:*filename* .

or

(b) *your_local_system*% scp *nas_username*@pfe*21*.nas.nasa.gov:*filename* .

Using shiftc

See Shift File Transfer Overview for shiftc examples.

More About File Transfers

See the list of links in the File Transfer Overview.

You can transfer files between NAS and your local site either by running the transfer command on a Pleiades or Lou front-end node (PFE or LFE), or by running the command on your local system. Running the command on your local system requires you to go through a secure front end (SFE) or to set up the Secure Unattended Proxy (SUP) or SSH passthrough. Therefore, if your local system is set up to allow direct inbound connections, then starting the transfer from a PFE or LFE will be much simpler than starting the transfer from your local system.

The sample command lines shown below demonstrate how to run the scp or shiftc commands on a PFE to transfer files to or from you local system. You can also apply the commands to other systems in the enclave, such as the LFEs.

Notes:

- The sample command lines use pfe21 as an example.
- Two command lines are provided for each transfer method. Use the first one if your username for your local system is the same as your username for the NAS systems. If the usernames are different, use the second.

Using scp

To push files *out* of a PFE to your local system:

pfe*21*% scp *filename your_local_system*:
pfe*21*% scp *filename local_username@your_local_system*:

To pull files *into* a PFE from your local system:

pfe*21*% scp *your_local_system:filename* .
pfe*21*% scp *local_username@your_local_system:filename* .

Using shiftc

In most cases, it is recommended to initiate remote transfers on the non-NAS side using SUP/Shift since all authentication, firewalls, and shift options have been preconfigured to work correctly. Remote transfers initiated from within NAS, however, are possible if the ssh port of the remote system is accessible and supports the publickey authentication method. The private key must either be loaded in your ssh-agent process or you must use a passphrase-less private key file with --identity set to its full path name. In addition, the file "shift-aux" (obtainable from /usr/local/bin on any pfe or lfe) must be in the default $PATH on the remote system. It is recommended to restrict the list of remote transports to rsync, fish, and shift unless you know which ports are accessible on the remote host and can specify --ports (for either fish-tcp or bbftp) and/or bbftpd is installed on the remote host (for bbftp).

To push files *out* of a PFE to your local system:

pfe*21*% shiftc --remote=rsync,fish,shift *filename your_local_system*:
pfe*21*% shiftc --user=*local_username* --identity=~/.ssh/*your_private_key filename your_local_system*:

To pull files *into* a PFE from your local system:

pfe*21*% shiftc --remote=rsync,fish,shift *your_local_system:filename* .
pfe*21*% shiftc --user=*local_username your_local_system:filename* .

See Shift Command Options for all available options. Transfers may be monitored and stopped/restarted using normal shift methods. See Checking Shift Transfer Status and Restarting Transfers for more information.

Optimizing/Troubleshooting

If you are moving large files, use the shiftc command instead of cp or scp. An online NAS service can help diagnose your remote network connection issues, and our network experts can work with your specific file transfer problems.

For fastest file transfer between Pleiades /nobackup and Lou, log into Lou and use shiftc, cxfscp, or mcp. A simple cp or tar will also work, but at slower speeds.

Moving large amounts of data efficiently to or from NAS across the network can be challenging. Often, minor system, software, or network configuration changes can increase network performance an order of magnitude or more.

If you are experiencing slow transfer rates, try these quick tips:

- Pleiades /nobackup are mounted on Lou, enabling disk-to-disk copying, which should give the highest transfer rates. You can use the shiftc, cp, or mcp commands to copy files or even make tar files directly from Pleiades /nobackup to your Lou home directory.
- If using the scp command, make sure you are using OpenSSH version 5 or later. Older versions of SSH have a hard limit on transfer rates and are not designed for WAN transfers. You can check your version of SSH by running the command ssh -V.
- For large files that are a gigabyte or larger, we recommend using shiftc. This application allows for transferring simultaneous streams of data and, when used without the --secure option, doesn't have the overhead associated with encrypting all the data (authentication is still encrypted).
- Another reliable option for large file transfers is through the Shift transfer tool, which includes options specific to the NAS environment, such as checking to see whether files residing on Lou are also on tape.

One-on-One Help

If you would like further assistance, contact the NAS Control Room at support@nas.nasa.gov, and a network expert will work with you or your local administrator one-on-one to identify methods for increasing your transfer rates.

To learn about other network-related support areas see End-to-End Networking Services.

On Lou, commands that should finish quickly may occasionally take a long time. This problem is usually due to slow retrieval of files from disk to tape.

When you run the ls command on Lou, the output shows all your Lou files on disk. However, most of the files are actually written to tape using the Data Migration Facility (DMF).

One reason for slow file retrieval is that for some multiple file transfers—for example, if you do an scp transfer with a list of files—Linux feeds each file to DMF one at a time, and DMF does not deal well with retrieving one file at a time from a long list of files. This means that the tape(s) containing the files is constantly being loaded and unloaded, which is very slow (and is bad for the tape and tape drives). As the list of files gets longer (through the use of "*" or moving a "tree" of files), the problem grows to where it can take hours to transfer a set of files that would only take a few minutes if they were on disk. This can be particularly problematic when several people do these types of file transfers at the same time.

The methods described below can help you avoid these problems.

Note: For more information about the commands in this section, see Data Migration Facility (DMF) Commands.

Optimizing File Retrieval

You can fetch files to disk as a group by running the dmget command before running your file transfer. dmget reads the tape once and gets all the requested files in a single pass.

Run dmget on the same list of files you are about to transfer. Then, after the dmget operation completes, you can transfer the files using scp/ftp/cp as you had originally intended. Or, you can put dmget in the background and run your transfer while dmget is working. If any files are already on disk, dmget sees this and doesn't try to get them from tape.

DMF also provides the dmfind command, which enables you to walk a file tree to find offline files to give to dmget.

Note: Be sure you are in the correct directory before running dmfind. Use the pwd command to determine your current directory.

Please check to make sure too much data isn't brought back online at once, either by using du with the --apparent-size option or by using /usr/local/bin/dmfdu. For example:

```
lou% /usr/local/bin/dmfdu filename
   filename
         13 MB regular          340 files
       1114 MB dual-state        1920 files
      74633 MB offline          2833 files
         13 MB small            340 files
      75761 MB total            5093 files
```

File transfer rates vary depending on the load on the system and how many users are transferring files at the same time. Typically, scp transfers between Lou and Pleiades on the /nobackup file system run between 30-120 MB/s for files larger than 100 MB, using the 10-gigabit network interface.

Example 1:

```
lou%  dmget *.data &
lou%  scp -qp *.data  myhost.jpl.nasa.gov:/home/user/dir_name
```

Example 2:

```
lou%  dmfind  /u/username/FY2000 -state OFL -print | dmget &
lou%  scp -rqp /u/username/FY2000 hostname:/nobackup/username/dir_name
```

You can see the state of a file by running dmls -l instead of ls -l.

Maximum Amount of Data to Retrieve Online

The online disk space for Lou is considerably smaller than its tape storage capacity, and it is impossible to retrieve all files to online storage at the same time. Using the Shift tool for file transfers automatically ensures that files on Lou are retrieved in batches and released afterwards so there is no need to manually split up the transfer. If you do not use Shift, however, then you should confirm whether there is enough disk space before you retrieve a large amount of data.

The df command shows the amount of free space in a filesystem. The Lou script dmfdu reports how much total (online and offline) data exists in a directory. To use dmfdu, simply cd into the directory you want to check, and execute the script.

If you would like to know the total amount of data under your home directory on Lou, you need to first find out if your account is under s1i-s1n or s2i-s2n. Assuming you are under s1c, you can then use dmfdu /s1c/*user_id* to find the total amount. Another alternative is to simply cd to your home directory and use dmfdu *, which will show use for each file or directory.

Lou's archive filesystems are between 85 TB and 450 TB in size, but the available space typically floats between 10% to 30%. In Example 3, 29% of space is unused.

It is best to retrieve no more than 10 TB at a time. As shown in Example 3, it is best to release the space (dmput -r) after using the retrieved files (scp, edit, compile, etc), then retrieve the next group of files, use them, and release the space again, and so on.

Example 3:

To retrieve one directory's data from tape, copy the data to a remote host, release the data blocks, and then retrieve more data from tape:

```
lou% df -lh .
Filesystem            Size  Used Avail Use% Mounted on
/dev/cxvm/sfa2-s2l    228T  196T  32T   86% /lou/s2l


lou% dmfdu project1 project2
project1
         2 MB regular            214 files
        13 MB dual-state           1 files
   2229603 MB offline            101 files
         2 MB small              214 files
   2229606 MB total              315 files

project2
         7 MB regular            245 files
      4661 MB dual-state          32 files
   2218999 MB offline             59 files
         7 MB small              245 files
   2223668 MB total              336 files


lou% cd project1

lou% dmfind . -state OFL -print | dmget &

lou% scp -rp /u/username/project1 remote_host:/nobackup/username

##(Verify that the data has successfully transferred)

lou% dmfind . -state DUL -print | dmput -rw

lou% df -lh .

lou% cd ../project2

lou% dmfind . -state OFL -print | dmget &

lou% scp -rpq /u/username/project2 remote_host:/nobackp/username

lou% dmfind . -state DUL -print  | dmput -rw
```

You can maximize your wide-area network bulk data transfer performance by tuning the TCP settings on your local host. This article shows some common configuration tasks for enabling high-performance data transfers on your system.

Note that making changes to your system should only be done by a lead system administrator or someone who is authorized to make changes.

Linux

1. Edit the file sysctl.conf located under the /etc directory, and add the following lines:

   ```
   net.core.wmem_max = 4194304
   net.core.rmem_max = 4194304
   ```

2. Then have them loaded by running `sysctl -p`

Windows

We recommend using a tool like Dr. TCP.

1. Set the "Tcp Receive Window" to at least 4000000
2. Turn on "Window Scaling," "Selective Acks,"and "Time Stamping"

Other options for tuning Windows XP TCP are the SG TCP Optimizer or using Windows Registry Editor to edit the registry, but the latter is only recommended for Windows users who are already familiar with registry parameters.

Mac OS 10.4

Note that these changes require root access.

In order to allow the Mac operating system to retain the parameters after a reboot, edit the following variables in /etc/sysctl.conf:

1. Set maximum TCP window sizes to 4 megabytes

   ```
   net.inet.tcp.sendspace= 4194304
   net.inet.tcp.recvspace= 4194304
   ```

2. Set maximum Socket Buffer sizes to 4 megabytes

   ```
   kern.ipc.maxsockbuf= 4194304
   ```

Mac OS 10.5 and Later

Use the sysctl command for the following variable:

```
sysctl -w net.inet.tcp.win_scale_factor=8
```

If you follow these steps and are still getting less than your expected throughput, please contact the NAS network group at support@nas.nasa.gov (attn: Networks). We will work with you on tuning your system to optimize file transfers.

You can also try the additional steps outlined in the related articles listed below.

Optional Advanced Tuning for Linux

This document describes additional TCP settings that can be tuned on high-performance Linux systems. This is intended for 10-Gigabit hosts, but can also be applied to 1-Gigabit hosts. The following steps should be taken in addition to the steps outlined in TCP Performance Tuning for WAN transfers.

Configure the following /etc/sysctl.conf settings for faster TCP

1. Set maximum TCP window sizes to 12 megabytes:

   ```
   net.core.rmem_max = 11960320
   net.core.wmem_max = 11960320
   ```

2. Set minimum, default, and maximum TCP buffer limits:

   ```
   net.ipv4.tcp_rmem = 4096 524288 11960320
   net.ipv4.tcp_wmem = 4096 524288 11960320
   ```

3. Set maximum network input buffer queue length:

   ```
   net.core.netdev_max_backlog = 30000
   ```

4. Disable caching of TCP congestion state (Linux Kernel version 2.6 *only*). Fixes a bug in some Linux stacks:

   ```
   net.ipv4.tcp_no_metrics_save = 1
   ```

5. Use the BIC TCP congestion control algorithm instead of the TCP Reno algorithm (Linux Kernel versions 2.6.8 to 2.6.18):

   ```
   net.ipv4.tcp_congestion_control = bic
   ```

6. Use the CUBIC TCP congestion control algorithm instead of the TCP Reno algorithm (Linux Kernel versions 2.6.18 and newer):

   ```
   net.ipv4.tcp_congestion_control = cubic
   ```

7. Set the following to 1 (should default to 1 on most systems):

   ```
   net.ipv4.tcp_window_scaling =1
   net.ipv4.tcp_timestamps = 1
   net.ipv4.tcp_sack = 1
   ```

A reboot will be needed for changes to /etc/sysctl.conf to take effect, or you can attempt to reload sysctl settings (as root) with `sysctl -p`.

For additional information visit the Energy Science Network website.

If you have a 10-Gb system or if you follow these steps and are still getting less than your expected throughput, please contact NAS Control Room staff at support@nas.nasa.gov, and we will work with you on tuning your system to optimize file transfers.

Streamlining PBS Job File Transfers to Lou

Some users prefer to streamline the storage of files (created during a job run) to Lou, within a PBS job. Because direct access to the Lou storage nodes from the Pleiades compute nodes and from Endeavour has been disabled, all file transfers to Lou within a PBS job must first go through one of the Pleiades front-end systems (PFEs).

Here is an example of what you can add to your PBS script to accomplish this:

1. ssh to a PFE (for example, pfe*21*) and create a directory on lou where the files are to be copied.

   ssh -q pfe*21* "ssh -q lou mkdir -p $SAVDIR"

   Here, $SAVDIR is assumed to have been defined earlier in the PBS script. Note the use of -q for quiet-mode, and double quotes so that shell variables are expanded prior to the ssh command being issued.

2. Use scp via a PFE to transfer the files.

   ssh -q pfe*21* "scp -q $RUNDIR/* lou:$SAVDIR"

   Here, $RUNDIR is assumed to have been defined earlier in the PBS script.

File Transfers Tips

The following quick and easy techniques may improve your performance rates when transferring files remotely to or from NAS.

This can increase your transfer rates by 5x, compared to older methods such as 3des.

- Transfer files from the /nobackup filesystem, which is often faster than the locally mounted disks.
- If you are using scp and your data is compressible, try adding the -C option to enable file compression, which can sometimes double your performance rates:

  % scp -C *filename user@remote_host.com*:

- For SCP transfers, use a low-process-overhead cipher such as aes128-gcm@openssh.com or arcfour:

  % scp -c -aes128-gcm@openssh.com *filename user@remote_host.com*:

- If you are transferring files from Lou, make sure they are online, rather than on the tape archive, before you perform the transfer operation.

  Note: If you use the shiftc command to transfer your files, it will automatically bring any files that are on the tape archive online before it transfers them. If you are not using shiftc, use the following DMF commands to determine the location of your files and bring them online if necessary:

  % dmls -al *filename* # show the status of your file.
  % dmget *filename* # retrieve your file from tape prior to transferring.

  For a full list of DMF commands, see DMF commands.

- If you are transferring many small files, try using the tar command to compress them into a single file prior to transfer. Copying one large file is faster than transferring many small files.
- For files larger than a gigabyte, we recommended using the Shift Transfer Tool, which can achieve much faster rates than single-stream applications such as scp or rsync.

To improve your performance by modifying your system, see TCP Performance Tuning for WAN Transfers.

If you continue experiencing slow transfers and want to work with a network engineer to help improve file transfers, please contact the NAS Control Room at support@nas.nasa.gov.

Troubleshooting SCP File Transfer Failure with Protocol Error

To address security issues with the scp command, we are in the process of adding checks to ensure that the files returned by a remote server match the files requested by the user. In some cases, the checks implemented in scp to address this issue may cause requested files to be rejected with the following error message:

```
protocol error: filename does not match request
```

This error can occur when you use quotation marks to escape special characters (such as a space) on the remote server, or when you use wildcard characters. The safest way to avoid this issue is to use the sftp command instead of scp to retrieve files. This avoids complications due to interpretation of the requested file names by the shell on the remote server.

For file retrieval, the syntax and command-line options for sftp are very similar to those for scp. For example, to retrieve files matching test*.c from a remote server to the directory *somedir*, use the following command line:

```
?$ sftp somehost:'test*.c' somedir/
```

Alternatively, you can add the -T option to the scp command line to disable checking of the file names returned by the remote server. Following the example above, the scp command line would be:

```
$ scp -T somehost:'test*.c' somedir/
```

If you need further help, please contact the NAS Control Room at (800) 331-8737 or (650) 604-4444.